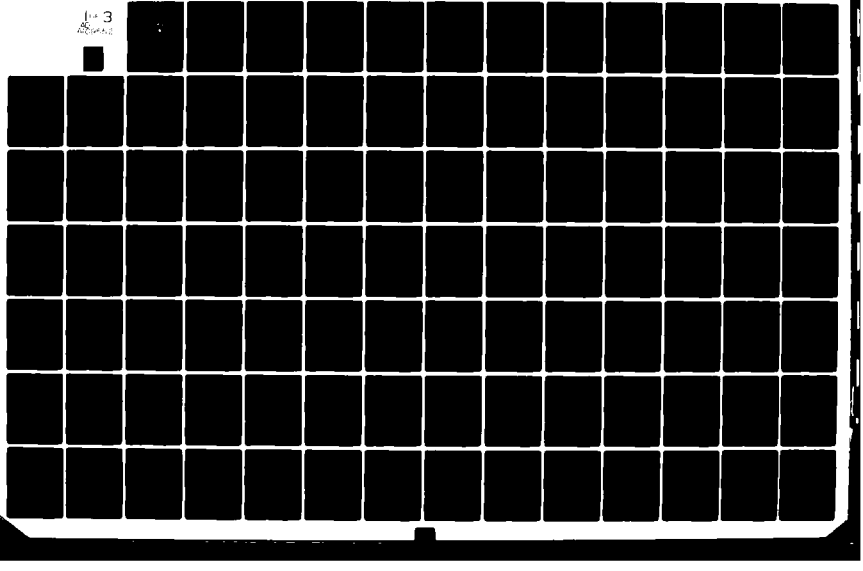
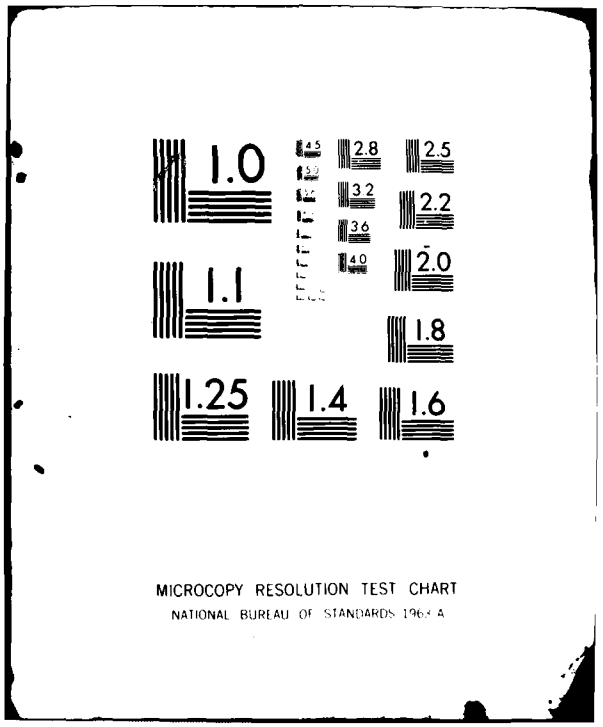
AD-A109 552 NAVAL POSTGRADUATE SCHOOL MONTEREY CA F/G 9/2
SASS HARDWARE ARCHITECTURE AND DEVELOPMENTAL MONITOR.(U)
JUN 81 G S BAKER
UNCLASSIFIED NPS52-81-016 NL




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963 A

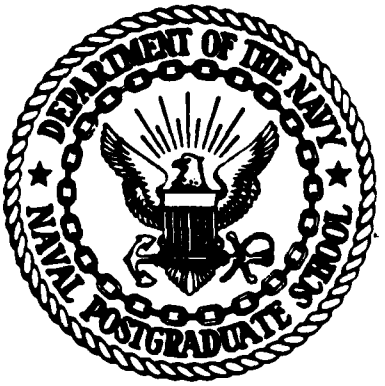
LEVEL II

(2)

NPS52-81-016

NAVAL POSTGRADUATE SCHOOL

Monterey, California



SASS Hardware Architecture
and
Developmental Monitor

Gary Stewart Baker

June 1981

Approved for public release; distribution unlimited

Prepared for:

Naval Postgraduate School
Monterey, CA 93940

NAVAL POSTGRADUATE SCHOOL
Monterey, California

Rear Admiral J. J. Ekelund
Superintendent

David A. Schrady
Acting Provost

The work reported here was supported by the Department of Computer Science.

This report was prepared by:

GARY STEWART BAKER
Lieutenant Commander, USNR

Reviewed by:

GORDON H. BRADLEY, Chairman
Department of Computer Science

Released by:

WILLIAM M. TOLLES
Dean of Research

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: NPS52-81-016	2. GOVT ACCESSION NO.: AD-109552
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): SASS Hardware Architecture and Development Monitor	5. TYPE OF REPORT & PERIOD COVERED: Technical Report
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): Gary Stewart Baker	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Naval Postgraduate School, Monterey, CA 93940	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: Naval Postgraduate School, Monterey, CA 93940	12. REPORT DATE: June 1981
	13. NUMBER OF PAGES: 278
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report): UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Secure Archival Storage System, Development Monitor, Secure Archival Storage System hardware architecture.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report contains the documentation for the Secure Archival Storage System hardware architecture, developmental monitor program, and all necessary support programs to effect programming of the firmware.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

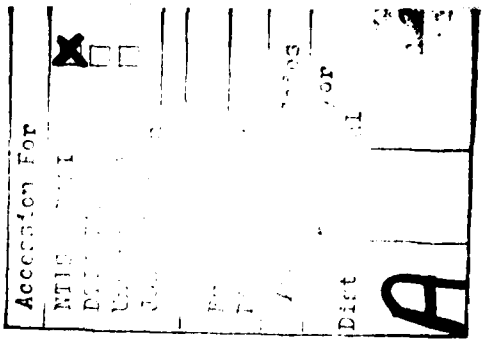


TABLE OF CONTENTS

LIST OF FIGURES

INTRODUCTION

This report contains the documentation for the Secure Archival Storage System hardware architecture, developmental monitor program, and all necessary support programs to effect programming of the firmware. The hardware architecture described in Appendix A provides the single board computer wire-wrap options and security modifications required to establish the hardware configuration. Appendix B presents

a command syntax tutorial for the monitor program and the program listing. The bootload program which comprises the current firmware is listed in Appendix C, with the support program and methodology for changing the firmware provided in Appendix E. The Bootstrap program, which loads and runs the SASS demonstration module, is presented in Appendix D.

The bootload program (firmware) has been tested in a single processor environment. The monitor program is currently being tested and debugged, but is considered usable as presented herein. The bootstrap program has not been sufficiently tested and will require further development.

APPENDIX A - Hardware Architecture

Presented in this appendix is an hardware resource desription meant to acquaint unfamiliar readers with the basic architectural devices and organization referred to in this thesis effort. The information necessary to reproduce the same hardware architecture is also contained in this appendix. Hardware familiarization includes the Zilog Z8000 microprocessor, the Zilog Z8010 Memory Management Unit (MMU), and the Advance Micro Computer Am96/4116 MonoBoard Computer. What follows is then a detailed description of the SASS Developmental Architecture built as a part of this thesis effort, to support the implementation of the initialization design presented in this thesis, and to support follow on work in the SASS. The intent was to make this appendix the hardware reference manual in support of future research efforts. Readers requiring more specific information on the hardware are referred to the appropriate literature listed in the references.

A. ZILOG Z8000 MICROPROCESSOR

The Z8000 is manufactured in two versions, the 48-pin Z8001 and the 40-pin Z8002, which differ only in the manner and range of memory addressing. Except for this feature, called memory segmentation, they are functionally identical. The Z8001 contains seven output lines for segment number

selection and one input line as a segment trap to support memory segmentation. Without segmentation, the Z8002 can address up to six distinct, external memory spaces of 64K bytes each, while the segmented Z8001 can access 128 addressable segments that are each 64K bytes in size. A total of 8 Megabytes can be addressed in this way with segmentation.

The CPU operates in one of two domains: system or normal mode. In the system mode all features of the processor are available to the running program, while in the normal mode the running program is isolated from potentially dangerous activities, such as instructions for basic I/O and changing system parameters.

Multiprogramming is supported by the concommitant Test-and-Set instruction and hardware context switching. Test-and-Set instructions allow single or block memory transactions without interruption, thus accommodating conventional spin-lock synchronization and concurrent read/write access to shared memory in a single processor environment. Changing the running environment or context of a process requires the reloading of the execution point which is defined in the Z8000 by two unique registers, the Program Counter (PC) and the Flag-and Control Word (FCW). The PC holds the address of the next instruction to be executed and the FCW contains the mode in which it is to be executed. Multiprocessing is facilitated by a combination of

instruction and hardware features. Bus control signals (BUSREQ and BUSAK pins) arbitrate the use of the multiplexed address and data bus by external devices, i.e. a DMA or disk controller. The Multi-Micro control signals (Micro-In and Micro-Out lines) when used in conjunction with certain special purpose instructions, allow a more general form of resource sharing among multiple processors.

In addition to the above required attributes, the Z8000 offers several very flexible architectural features that can be classified as: CPU control and status, register structure, and addressing modes.

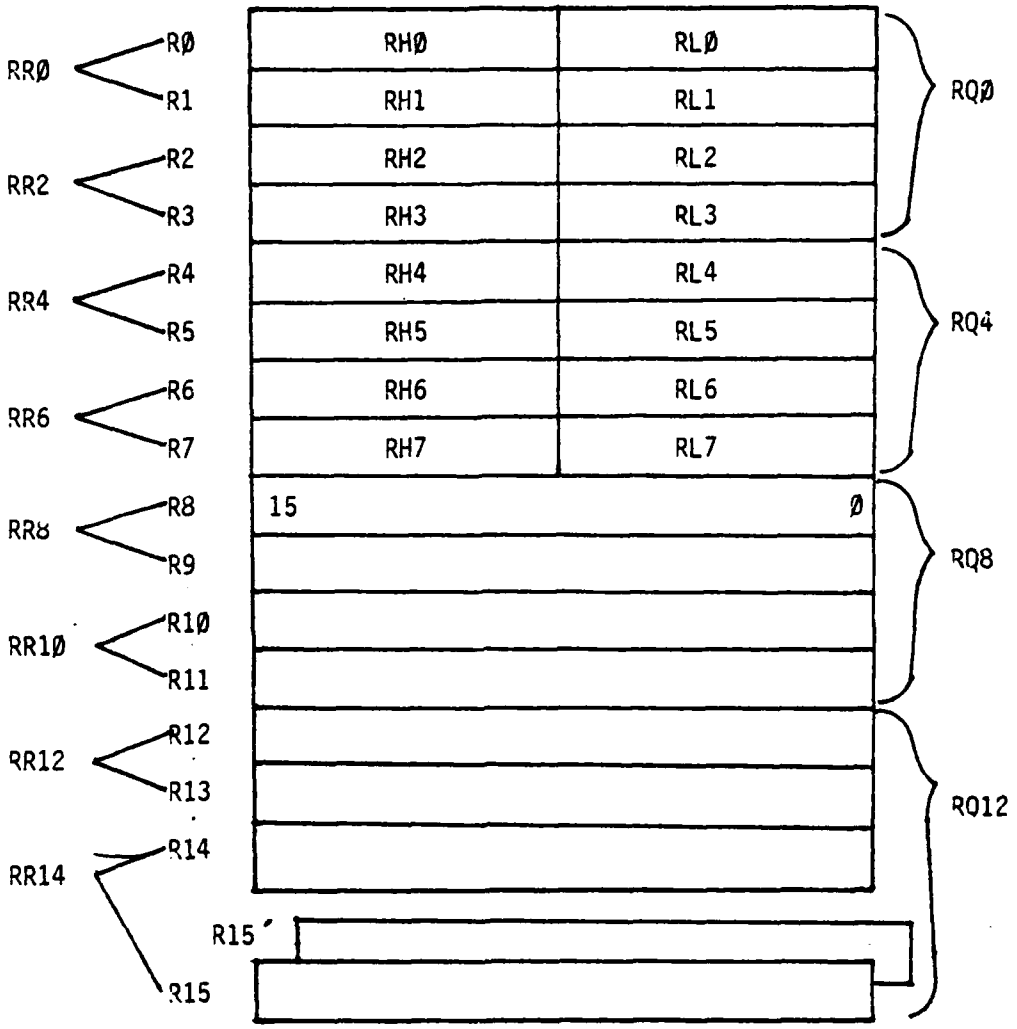
1. Register Structure

The Z8000 CPU is a register-oriented machine that provides a regular register structure for manipulating bytes (8-bits), words (16-bits), and long-word (32-bits) values. The instruction set provides for bit and nibble (4-bit) access to the various register contents. The sixteen 16-bit registers in the architecture can be referenced as sixteen 8-bit, sixteen 16-bit, eight 32-bit, or four quad-word 64-bit registers. All word or long-word registers (with the exception of R0 and RR0) may be used as general purpose stack registers and manipulated with the PUSH and POP instructions. The R15 and RR14 registers are the stack pointers used by the Z8002 and Z8001 respectfully, when executing CALL and RETurn instructions, and when processing interrupts and traps. Figure A-1 shows the register

structure of the Z8002 processors. The Z8001 requires a R14' register for segmentation.

Additional special registers also comprise the register architecture. As described previously, the execution point of a process is contained in two unique registers, the Program Counter (PC) and the Flag-and Control Word (FCW). The high byte of the FCW sets the mode of operation of the processor by selecting normal/system, segmented/non-segmented, and by enabling or disabling vectored interrupts (VI) and non-vectored interrupt (NVI). The lower byte of the FCW contains flags that may be used by any program to make conditional jumps or other control decisions affecting program flow of control. These flags are set or reset depending on the results of preceding instructions. Figure A-2 illustrates the organization of the program status registers.

Exception processing is directed through the use of another special register, the Program Status Area Pointer (PSAP). At initialization a table in memory must be allocated which contains all the FCW and PC values needed for each of the possible internal (traps) and external (interrupts) events. The organization of this Program Status Area pointed to by the PSAP is shown in Figure A-3. Interrupts are external asynchronous events requiring CPU attention, generally by external devices. Traps are synchronous events resulting from the execution of certain



Z8002 General Registers

FIGURE A-1

instructions. Both are processed in a similar manner by the CPU. The processor supports three types of interrupts: non-maskable, vectored and non-vectored; and four system traps: system call, unimplemented instruction, privileged instruction and segmentation trap. The vectored and non-vectored interrupts are internally maskable within the CPU. When an exception occurs, the current program status is automatically pushed onto the system stack. The program status in this instance refers to the processor status (PC and FCW) and a 16-bit identifier containing the reason or source of the exception.

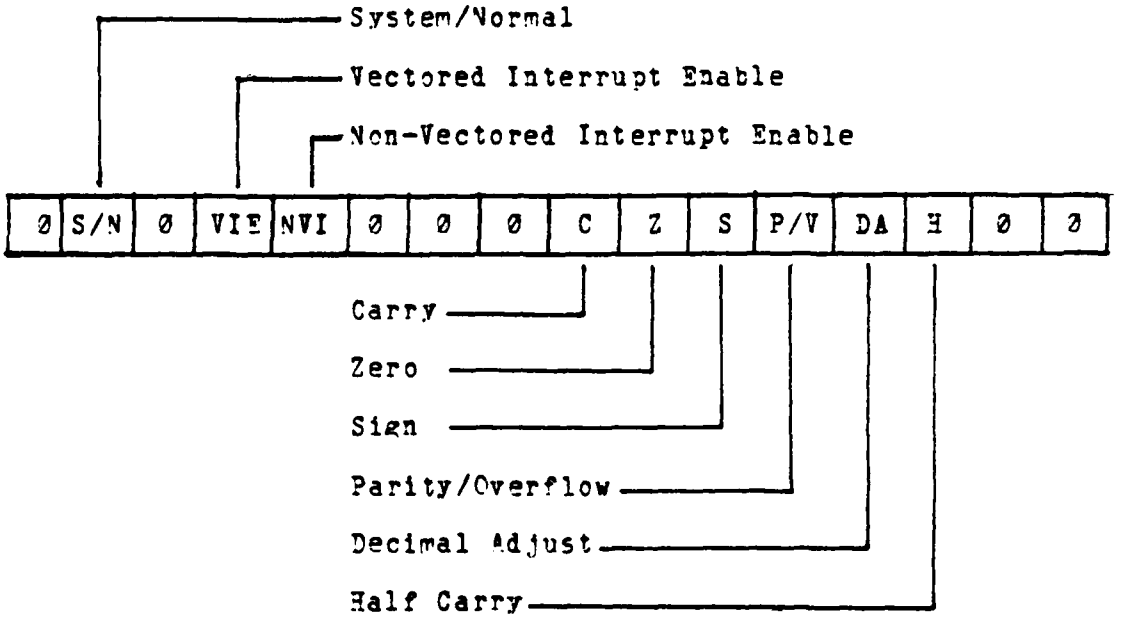
The last special register in the register structure is the REFRESH register, which is designed to provide dynamic memory refreshing on an estimated demand basis. The upper byte is loaded with a refresh rate count and bit-15 is set to enable the memory refresh activity.

2. CPU Control and Status

The Z8000 CPU incorporates three control input lines and four status output lines. Input control lines change CPU states, while the status output lines may be decoded to indicate internal CPU status (e.g., instruction fetches, I/O references or interrupt acknowledges).

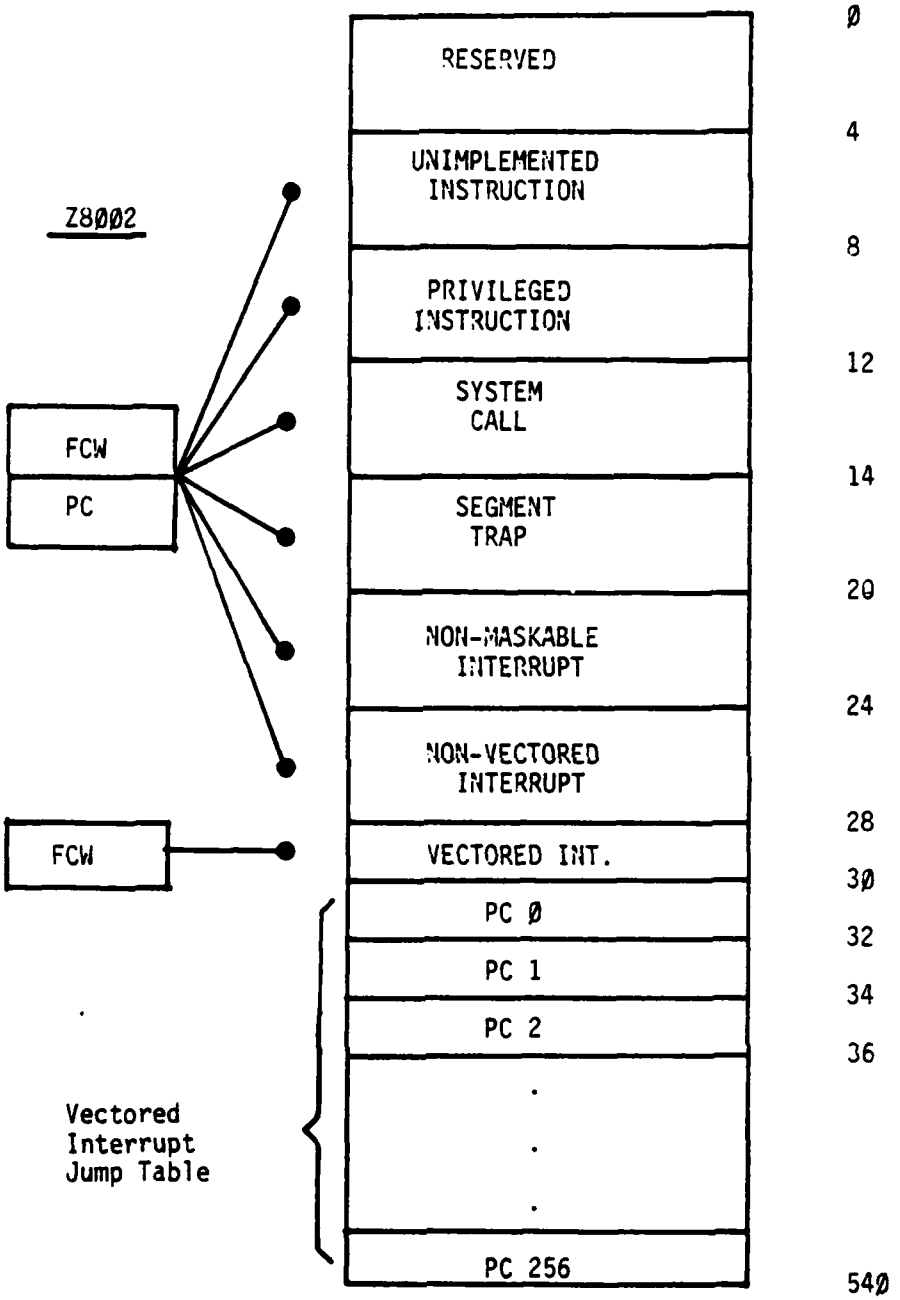
The RESET control line interrupts normal CPU operations and causes the CPU to clear itself completely (even of interrupt and bus requests). Two consecutive read cycles are executed in the system mode that load the FCW

Flag-and Control Word



Program Counter

Z8002 Program Status (PS) Registers
FIGURE A-2



Program Status Area (PSA)

FIGURE A-3

with the contents of memory location 0002 and load the contents of location 0004 into the PC. In the case of the segmented Z8001, location 0004 contains the PC segment number and 0006 contains the offset within the segment. In either case, after loading of the initialized system execution point, the following first instruction fetch (IF1) starts the CPU running. This first PC must, of course, point into executable code, typically in firmware.

For special purposes, external devices can control the execution of the CPU by activating the STOP control line. This freezes the next instruction without loss of memory refresh as occurs with bus requests. This control line can be used to synchronize the interactions of a Z8000 and extended-processor units (EPU's), which can, for example perform floating point operations.

The WAIT signal input will allow slower external devices to stretch the number of clock cycles between the address strobe (AS) and data strobe (DS) within a machine cycle, for as long as necessary to receive or assemble the data. In this way, a 16-bit address directs a 16-bit data transfer that is essentially independent of the processor's clock and is thus asynchronous.

Internal CPU status indications can be obtained by decoding the four output status lines from the processor. Figure A-4 lists the decoded signals. Decoded status, for example can be used to segregate program, data and stack

ST 3:0				CPU STATUS
0	0	0	0	Internal Operations
0	0	0	1	Memory Refresh
0	0	1	0	I/O Reference
0	0	1	1	MMU I/O Reference
0	1	0	0	Segment Trap Acknowledge
0	1	0	1	Non-Maskable Int Acknowledge
0	1	1	0	Non-Vectored Int Acknowledge
0	1	1	1	Vectored Int Acknowledge
1	0	0	0	Memory Reference - Data
1	0	0	1	Memory Reference - Stack
1	0	1	0	N/A
1	0	1	1	N/A
1	1	0	0	Memory Reference - Code (IFn)
1	1	0	1	Memory Reference - Code (IF1)
1	1	1	0	N/A
1	1	1	1	N/A

CPU Status Lines
FIGURE A-4

memory areas. When used in conjunction with the normal/system output line from the CPU, six distinct physical memory regions can be formed: normal program, data, and stack; and system program, data, and stack. The four status output lines allow the CPU many flexible control options over external devices.

3. Adressing Modes

The addressing mode of a Z8000 instruction defines, either explicitly or implicitly, the address space it references and the method used to compute the address itself. In general, an addressing mode explicitly specifies either register address space or memory address space. Program memory address space and I/O address spaces are usually implied by the instruction. Data may be addressed in eight basic modes: Register (R), Immediate (IM), Direct Address (DA), Indirect Register (IR), Indexed (X), Relative Address (RA), Based Address (BA) and Base Indexed (BX).

In the Immediate mode (IM), the data is part of an instruction itself; in the Register mode (R), the register to which the operand is to be written into or read from is specified. The address of the operand is carried directly with the instruction in the Direct Adressing mode (DA). The IM, R and DA modes all require that the operand's address be static or known at compile time, before the program is run.

The remaining address modes allow dynamic or runtime computation of addresses and/or displacements in registers. The Indirect Register mode assumes the address of the operand to be placed at runtime in the specified register contained in the instruction; the Indexed mode allows a program to calculated a displacement from a fixed base address. This is more commonly used for indexing fixed tables. The Base Address mode is a reflection of the indexed mode in that the base address rather than the displacement is calculated at runtime. A common use of this mode is for accessing identical portions of different instances of a data structure. The Base Indexed mode combines the base address and indexed modes to allow the creation of fully relocatable, reentrant code (relocatable specifies that the code may be moved to any location in memory, and reentrant means that the code does not modify itself). The final addressing mode is the Relative mode, inwhich the PC is always used as the base address, combined with the specified displacement contained in the instruction, to determine the absolute address.

B. ZILOG Z8010 MEMORY MANAGEMENT UNIT

In the segmented Z8001 processor, the addresses actually manipulated are logical two-dimensional addresses consisting of a 7-bit segment number and a 16-bit offset within the segment. The Zilog Z8010 Memory Management Unit (MMU) takes the 23-bit logical addresses from the CPU and transforms

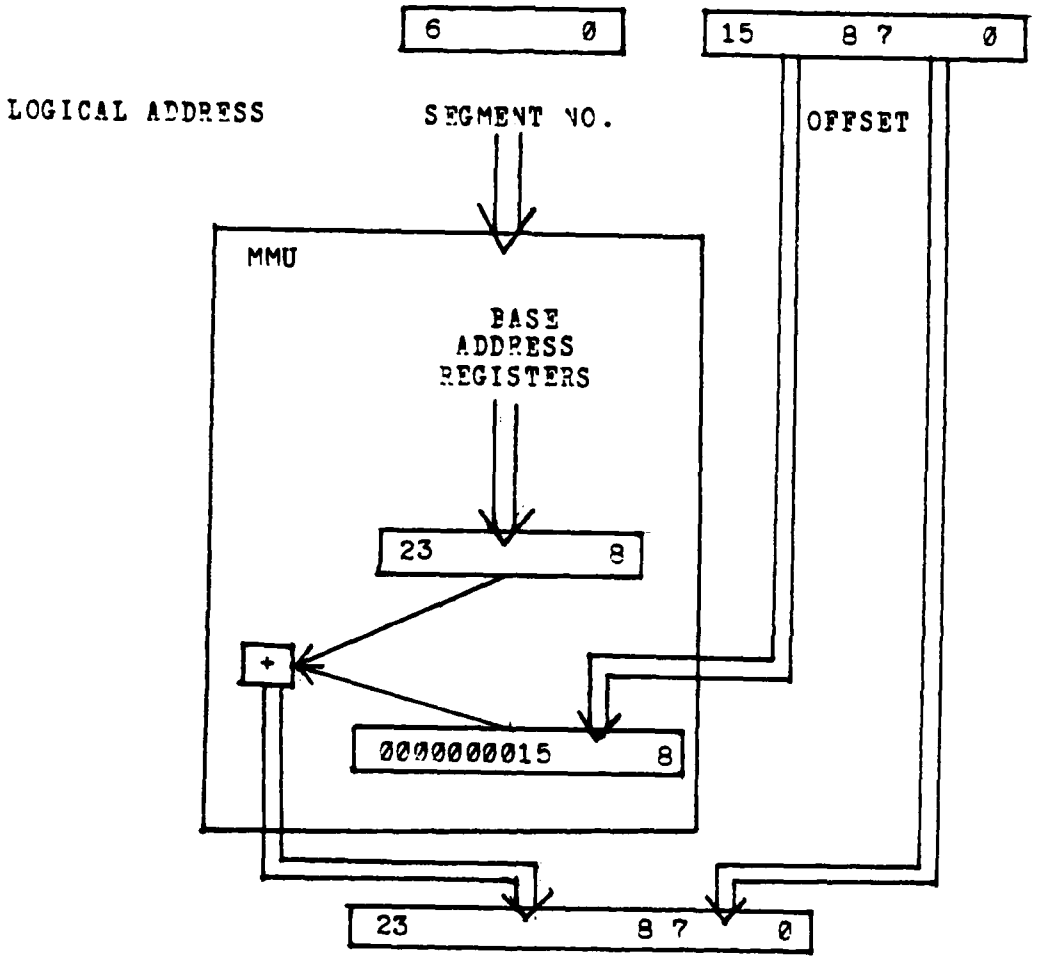
them into 24-bit absolute addresses for addressing physical memory[22]. This address transformation process is called relocation. A translation table of registers in the MMU associates the 7-bit segment number (SN) with the base address of one of 64 diferent physical memory segments. The 16-bit offset is added to the physical base address to obtain the actual physical address. Figure A-5 illustrates the relocation process. The base address register file may be dynamically reloaded as processes are created, suspended or changed.

The MMU also provides segment protection and memory management facilities. Each segment can have several attributes associated with it that provide memory access protection (viz. read/write access, normal/system mode only) and memory management data (viz. whether changed or referenced).

MMU defined segments are variable in size ranging from 256 bytes to 64K bytes in increments of 256 bytes. Pairs of 64-segment MMU's are necessary to support the 128 segment numbers available from segment number line decodings.

C. AMD Am96/4116 MONOBCARD COMPUTER

A major factor in the selection of the Zilog Z8000 family of microprocessors was the commercial availability of the Advanced Micro Devices (AMD) Am96/4116 MonoBoard Computer (MBC). The MBC is a complete microcomputer system

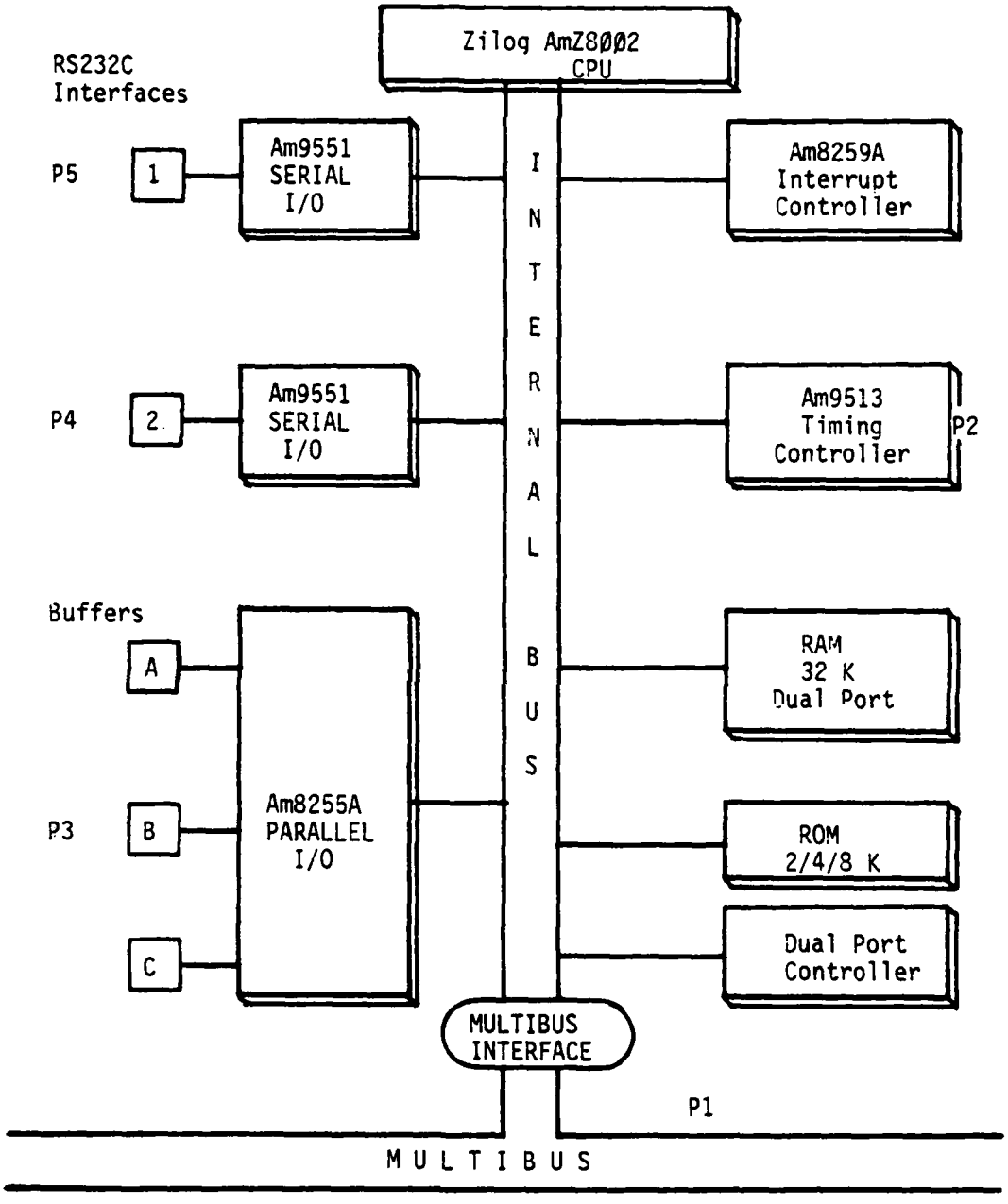


MMU Address Relocation
FIGURE A-5

on one 12.0 by 6.75 inch printed, multi-layered circuit board. The MBC version available at the outset of the SASS project contains a 5Mhz Z8002 non-segmented processor, 32K bytes of random-access memory (RAM), up to 8K bytes of electronically programmable read-only memory (EPROM), programmable serial and parallel I/O interfaces, a real-time clock and counter system, a programmable interrupt controller, and five edge connectors providing bus and peripheral interface capability [23]. Figure A-6 depicts the MBC architecture in block diagram form.

Multibus interfacing is through standard edge connector P1, which supports 20-bit addressing, eight bus interrupt lines, and bus arbitration and timing lines. The Am9551 serial I/O components interface offboard to peripheral devices through edge connectors P4 and P5. Serial port 1 (P5) supports all interconnections of a standard RS232C interface, while serial port 2 (P4) does not support synchronous handshaking capabilities. Edge connector P3 interfaces the three 8-bit parallel ports (A,B,C), from the Am8255A parallel I/O integrated circuit. All of the inputs and outputs of the Am9513 Timing Controller chip are available through edge connector P2.

Included on the Am96/4116 is a 32 source interrupt jumper matrix for input to any of the eight interrupt inputs (IRQ0 - IRQ7) of the Am8259A Interrupt Controller. The jumper matrix contains bi-directional Multibus interrupt



Zilog Am96/4116 MonoBoard Computer

FIGURE A-6

interfacing (INT0* - INT7*) for external device interrupt communications; two parallel port (C) single bit sources for use as software generated interrupts; receive and transmit ready signals from the two serial I/O components for asynchronous communications with peripheral devices; five timer or counter output signals from the Am9513 IC; and several sources from the onboard fault detection circuitry. All interrupt sources are available as a Z8000 non-vectored interrupt or non-maskable interrupt as well, through the interrupt jumper matrix.

The Am8255A parallel I/O component provides three 8-bit parallel ports, one of which (C) can be used to program under software control, the function of the other two ports (A and B). In addition, a 4-bit portion of port C can be used as a Multibus address source for the higher address bits (AD10-AD13), under program control. The significance of this will be seen later on. Two bits of the C port, as mentioned previously, also serve as signal sources for software generated interrupts.

The onboard dynamic RAM (32K bytes) is dual ported, with Multibus access controlled by an onboard control logic. This capability is designed for a multi-processor environment where maximum flexibility of available physical memory is desired; however, the SASS design goal of memory segmentation cannot tolerate a single RAM memory space having two addresses and thereby requiring particular care

to disable this option. The MBC can support an optional 2, 4, or 8K byte read-only memory (ROM) in a shadow mode; the ROM is shadowed over existing memory space by CPU direction. The ROM can support the system firmware.

Provided on the MBC is an Am9513 Timing Controller that contains five Counter Logic Groups consisting of a 16-bit general counter with associated control and output logic, a 16-bit Load register, a 16-bit Hold register and a 16-bit Mode register [20]. In addition, counter groups 1 and 2 also include 16-bit comparator and Alarm registers. The Counter Mode Register is used to control all of the individual options available with its associated general counter group. The Load register is used to control the effective length of the general counter, while the Hold register is used to store accumulated counter values for later transfer to the host processor. The Alarm and comparator registers are used to detect and signal specific values in the general counter. Each of the counter groups can serve as independent or cascaded, count-down or count-up counters with gated or nongated count sources. Each provides an output signal to the interrupt jumper matrix and the P2 edge connector for offboard applications.

CPU onboard accessing of the individual IC's is accomplished through a decoding of the addresses contained in the Z8000 IN/OUT instructions; components are enabled appropriately. All programmable components must be

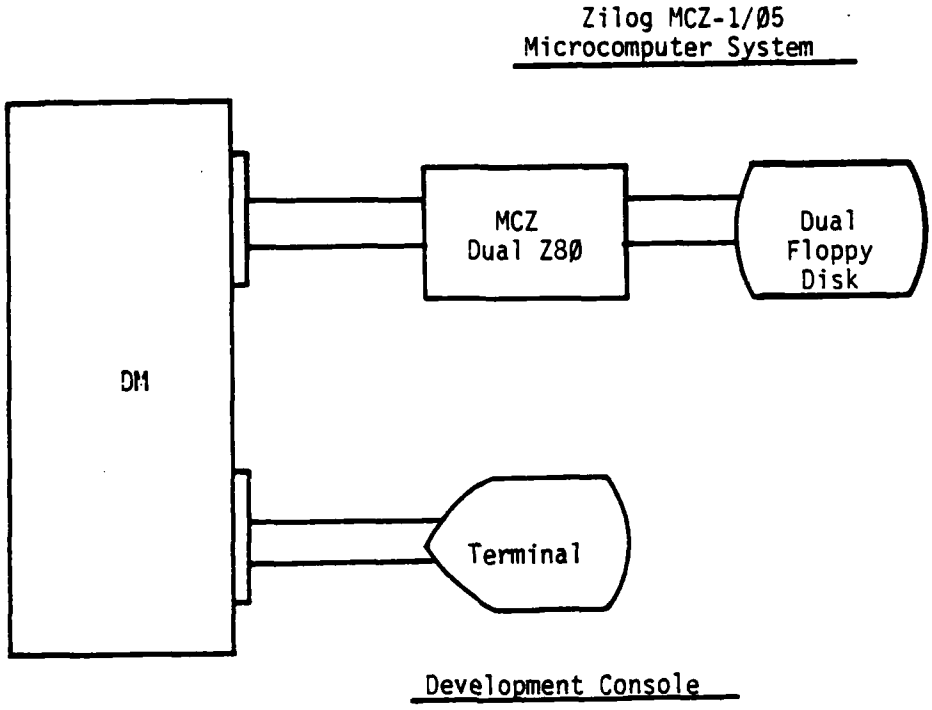
initialized to some consistent state as the first step in any initialization mechanism.

D. DEVELOPMENTAL SYSTEM

As stated previously, the primary feature desired in the SASS Developmental Architecture (SDA) is the external bus. To acheive this, the Am96/4116 Monoboard must replace the Zilog Z8000 Development Module (DM) which has previously been the platform for SASS development. Figure A-7 shows the end product after the substitution. The remaining developmental components from the original system consist of the Zilog MCZ-1/05 Microcomputer System (MCZ) and a CRT terminal. A firmware monitor program for the MonoBoard was needed to affect the substitution. The firmware will be discussed under the SDA to follow.

1. The Zilog MCZ-1/05 Microcomputer System

The MCZ-1/05 Microcomputer System is a disk-based microcomputer unit, utilizing the Z80 family of microcomputer cards. The basic system includes two boards: a Z80 microcomputer board (MCB) for performing primary system control, and a Z80 memory disk controller (MDC) for interfacing the dual floppy disk drive. These two cards constitute the basic MCZ system, that provides 60K bytes of RAM memory, 3K bytes of firmware, control of the disk drive module, and communications with the Z8000 MBC through an asynchronous serial RS-232 interface.



Z8ØØØ Development System

FIGURE A-7

Supporting software consists of a full-disk operating system (RIO), which includes a macroassembler, linker, text editor, and file utility. Details concerning the MCZ hardware or software can be found in the listed references. The macroassembler provides for assembling relocatable object modules from PLZ/ASM language source code entered with the RIO text editor. Higher level language support, as PLZ/SYS, is currently not available for developmental use. Subsequently relocatable modules can be linked together with the Linker and core images located with the Imager. Object files are then downloaded into the MBC memory for executing and debugging. The ability to upload from the MBC memory into the RIO file structure is also available.

User communication with the RIO operating system is via the development terminal, through the Am96/4116 MonoBoard. The MBC developmental monitor provides for three modes of operation: 1) the Monitor Mode, which is used to enter, debug and execute software residing in MBC RAM; 2) the Transparent Mode, which is used to transmit data between the development terminal and the MCZ system; and 3) the Upload/Download Mode, used for transferring object files between MBC memory and the MCZ microcomputer system. Offboard Am96/4116 MonoBoard communications with the MCZ system and the development terminal is asynchronous through the two MBC serial ports as designated in Figure A-7.

2. Am96/4116 Configuration

The commercial MonoBoard provides wire-wrap options for configuring the MBC for specific applications. The options available include primary memory addressing (both onboard and offboard), interrupt/trap source selection, and various device configuration settings. An appropriate wire-wrap configuration for each of these areas was defined from requirement consideration of the SASS and the developmental system. Figure A-8 lists the finalized MonoBoard wire-wrap configuration. This configuration results in the following interrupt utilization in the SASS Developmental Architecture:

Non-Vectored Interrupt - Hardware PRE-EMPT
Vectored Interrupts:

INT0	-	Console INT
INT1	-	N/A
INT2	-	MCZ INT
INT3	-	Onboard PRE-EMPT
INT4	-	N/A
INT5	-	Real Time Clock
INT6	-	Single Inst. Execution
INT7	-	Illegal Memory Reference

Interrupt handler routines when assigned in this manner, are supported by this wire-wrap configuration.

For compatability with the Am96/4116 RAM Memory Board, address bits AD8 and AD10 for offboard addressing were switched. Address bit AD8 uses the buffered Normal/System CPU signal as its source. When the secondary storage device (i.e., hard disk) is installed, offboard address bit AD11 should be sourced to the instruction fetch

signal (I-FETCH*) signal by connections of pins 68-69 and 38-39. This will allow code execution only from one of the two memory boards, and data fetches in two domains within the other.

3. Am96/4116 Modifications

Global memory partitioning by domains is accomplished by making use of the processor's Normal/System signal for offboard addresses. Partitioning of onboard local memory is more difficult. Use of available onboard logic gates and wiring modifications were required to achieve the desired segregation. In the SASS design for MMU simulation, a portion of local memory must be accessible in the system mode only, thereby protecting the information contained in this area (i.e., the Kernel). For simplicity the design choice was made to use the address bit ABE to equally divide the local memory space. The desire is to make the lower half (0000-3FFF) accessible in the system mode only and the upper half (4000-7FFF) accessible in both the normal and system mode. Given those two signals, the following truth table was derived:

POWERUP RESET/INIT
Pin 190 - 191

BUS CLOCK SELECT
Pin 19 - 20

DAISEY-CHAIN BUS PRIORITY
Pin 184 - 185 (no chain)
21 - 22

SERIAL PORTS 4 & 5
Pin 132 - 133 (RTS1-CTS1)
139 - 140 (9600 BAUD)
141 - 143 (9600 BAUD)
179 - 180 (CTS2-GND)

ONBOARD RAM SELECT
Pin 193 - 194
(0000-7FFF)

GLOBAL MEMORY SELECT
Pin 36 - 63 (ADF-AD10)
62 - 37 (N/S-ADF)
65 - 66 (N/S)

PROM SELECTION (2716)
Pin 25 - 27
28 - 30
177 - 178
174 - 176
71 - 72

OTHER

Pin 33 - 34	Pin 136 - 137
38 - 39	148 - 162
40 - 41	151 - 150
42 - 43	153 - 154
128 - 130	163 - 164

INTERRUPTS

Vectored Interrupts:

(SSTPP)	Pin 130 - 115	(IREQ06)	"Next" command
(OUT2)	114 - 117	(IREQ05)	Real time clock
(RXR2)	100 - 127	(IREQ00)	Console Interrupt
(RXR1)	101 - 123	(IREQ02)	MCZ Sys Interrupt
(OUT4)	116 - 95	(INT4*)	Multibus Preempt
(OUT5)	112 - 121	(IREQ03)	Onboard Preempt
(Spec Mod)	* - 113	(IREQ07)	Memory Violation

Non-Maskable Interrupts:

(INT1*)	Pin 79 - 128	(NMI)
(Timeout)	157 - 171	(NMI)
(Oddword)	158 - 172	(NMI)

Non-Vectored Interrupt:

(INT4*)	Pin 95 - 129	(NVI)	Hardware Preempt

Am96/4116 MBC Wire-Wrap Options
FIGURE A-8

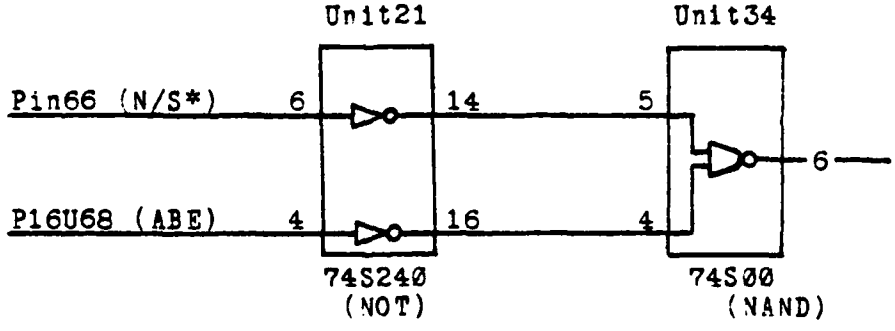
N/S*	ADE	^N/S*	^ADE	+	NAND	
1	1	Ø	Ø		1	NORMAL
1	Ø	Ø	1		1	
Ø	1	1	Ø		1	SYSTEM
Ø	Ø	1	1		Ø	

Address Bit Truth Table

Figure A-9

The NAND signal becomes the new ABE bit.

A search of available onboard logic gates produced the necessary components to construct the circuitry, and solder connections were made as shown:



wire:	Pin66	to	P6U21
	P14U21	to	P5U34
	P16U68	to	P4U21
	P16U21	to	P4U34
	P6U34	to	P13U66
cut:	P16U68	to	P13U66

Address Bit Modification

Figure A-10

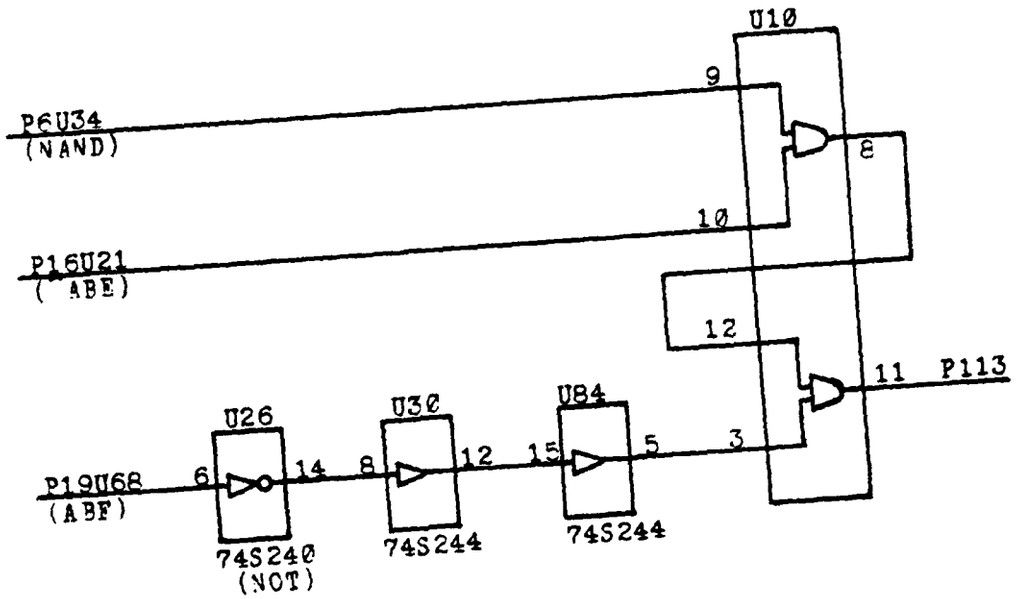
The above modification produced the desired security, however addressing in the normal mode below 4000 HEX produced some undesirable effects. For instance, a write to address 2000 HEX would actually write to address 6000 HEX, unknown to the user. Another design decision was made to prevent this occurence by providing an interrupt source from this circuitry. The following logic generated the interrupt:

ABE	~ABE	'NAND'	+	'AND'
1	Ø	1		Ø
Ø	1	1		1
1	Ø	Ø		Ø
Ø	1	Ø		Ø

Interrupt Truth Table

Figure A-11

However, precautions must be taken to preclude the generation of this interrupt when addressing global memory from 8000-C000 HEX. The address bit ABF was used to enable/disable interrupt generation in the following circuitry:



Interrupt Modification
Figure A-12

The additional drivers are supplied to the ABF signal for timing considerations. With this circuitry an interrupt (INT7) is generated only when an access to low local memory (0000-3FFF) is made in the normal mode.

4. Am96/1064 Memory Board Configuration

The Am96/1064 is a single board dynamic random access memory system with 64K bytes of addressable memory and onboard control and refresh logic for maintaining stored data. It is Multibus compatible with a 415nsec access time. Normal installation procedures were adhered to with the following execeptions. On one of two such boards in the system, bus address comparator (U111) is allowed to pass

address bit AD10 (N/S*) to onboard logic for selection of one of two planes of 32K bytes of memory. One addressable in the normal mode and the other addressable in the system mode.

As mentioned earlier, when a suitable secondary storage device is installed the second memory board can be added and the instruction fetch signal (I-FETCH*) can become the source for address bit AD11. For the memory board mentioned above, address bit AD11 will be forced high (Pin connections 60-59 removed) and the second board will have this bit pulled low (I-FETCH*) on the comparator. This provides the read-only function. No additional wiring modifications are required.

C. SUMMARY

This section provided a brief tutorial into the architecture of the Z8000 family of microprocessors. The intent was to provide the reader with the necessary hardware background to understand the content of this thesis, and to reconstruct the hardware architecture used in this effort. In answer to any additional questions that may arise, the reader is encouraged to consult the referenced literature.

APPENDIX B - Developmental Monitor Program Listing

The developmental monitor program is presented in this appendix in two parts, the command syntax description and program listing. The command syntax description is a complete users manual for the monitor program. The program listing contains the eight program modules that comprise the PLZ/ASM program listing. The somewhat unstructured program appearance stems from the requirement to maintain the organizational structure of certain required functions, in particular the MCZ system communication protocol. A conscious effort was made to preserve the original command syntax.

A. COMMAND SYNTAX

The developmental monitor consists of eleven commands: four performing normal development functions for display and setting of processor memory and registers; four operations for debugging of user programs; two providing upload/download capability with the MCZ system; and one command for entering a transparent mode of operation between the MCZ system and the user console. A command line description is presented for each command. Command line representations adhere to the following conventions:

(1) the use of angle brackets <> denotes a required entry,
(2) the use of square brackets [] denotes an optional entry.

The command lines serve as a ready reference for command syntax familiarization in the following descriptions.

1. DISPLAY Command

D[isplay] <adr> [no. words]

The display command can be used to perform two operations: the display of a block of memory in word format (16-bits), and to display the contents of a single address with the option to change the contents. Use of the command with a starting HEX address, followed by a hexadecimal number indicating the number of words desired, will display the contents of a block of memory. Use of the command followed by a single address enters a substitution mode. The contents of the single address will be displayed on the console followed by a prompt (*) indicating to the user that he has three options: (1) entering a new value will change the contents of that location, (2) entering a carriage return <CR> sequentially steps through memory leaving the contents unchanged, and (3) entering a 'Q' will terminate the substitution mode of operation and return control to the monitor program.

2. REGISTER Command

R[egister] [register name]

The register command displays the current contents of registers RØ-R15, PC and FCW. Use of the command alone will display the current contents of all registers and return to the monitor. Use of the optional register name will enter a substitution mode as described above, and proceed sequentially until either a 'Q' is entered or all registers have been displayed. Control is then returned to the monitor.

3. MOVE Command

M[ove] <old adr> <new adr> <no. bytes>

The move command moves the contents of the specified block of memory to a new location, without altering the contents of the old locations. The old address and number of bytes specifies the block to be moved; and the new address specifies the new starting address.

4. FILL Command

F[ill] <start adr> <end adr> <data>

The fill command changes the memory contents inclusively from the specified starting address to the ending address, with the user supplied hexidecimal data.

5. GO Command

G[o]

The go command starts user program execution at the execution point defined by previously user set PC and FCW registers.

6. JUMP Command

J[ump] <start adr>

The jump command starts execution of a user's program from the address specified, using the current FCW register value.

7. BREAK Command

B[reak] [address] [no. breaks]

The break command is used to set a break point within a user's program. When the break point is encountered during normal execution of the user program, execution is terminated, user registers and program status is saved, and the message 'BREAK AT <address>' is sent to the console. Use of the break command alone will clear any set break points. User has the option of specifying the number of times that the break point must be encountered before program execution is terminated; the default value is 1.

8. NEXT Command

N[ext] [no. instructions]

The next command is used to single step through the execution of a user's program. The contents of the CPU registers after each instruction execution are displayed to the user. Multiple instruction executions can be optionally entered by the user; the default value is 1.

9. QUIT Command

Q[uit]

Use of the quit command places the MonoBoard in a transparent mode where it performs as a communication relay between the MCZ RIC operating system and the user console. To the user the appearance is complete operation within the MCZ system. To exit the transparent mode, the Non-Maskable Interrupt (INTR switch) must be given, which saves user registers and returns control to the monitor.

10. LOAD Command

L[oad] <filename> [load adr]

The load command downloads program files from the MCZ RIC file structure into the Z8000 system for execution. The load address can be optionally specified; default is to the load address pass' with the file from RIC. After a

successful download operation, the message 'ENTRY POINT <address>' is sent to the console showing the load address used.

11. SEND Command

S[end] <filename> <start adr> <end adr> [entry adr]

The send command is used to save the contents of a specified block of Z8000 memory within the RIO file system. The optional entry address becomes the default load address for downloading with the load command; otherwise the start address becomes the default address.

12. Alerts

Within the hardware architecture are certain interrupts that require user notification on occurrence. The following is a list of the current interrupt messages:

'NMI'	Results from a Non-Maskable interrupt generation by depression of INTR switch.
'ILLEGAL MEMORY REFERENCE'	Results from illegal memory reference in normal mode.
'UNKNOWN TRAP'	Results from interrupt/trap not currently used in the architecture.

What follows in the next section is an annotated source listing for the monitor program.

B. MONITOR PROGRAM LISTING

1. EXECUTIVE MODULE

```
Z8000ASM  2.02
LOC     OBJ CODE     STMT SOURCE STATEMENT

                       1 EXEC_DMONITOR MODULE
                $LISTON  $TTY

                CONSTANT

                   RXR        :=  2
                   TXR        :=  0
                   PAR        :=  7
                   PORTAD     :=  %FFD9
                   PORTBD     :=  %FFE1
                   PORTAC     :=  %FFDB
                   PORTBC     :=  %FFE3

                   IDPORT     :=  %FFCB
                   ICPORT     :=  %FFC9

                   TCMD    :=  %FFD2
                   TDTA    :=  %FFD0

                   BUS_LOCK :=%FFF9
                   BUS_UNLOCK :=%FFF8
                   VINTR      :=  %(2)0001000070000000
                   VIBIT      :=  12
                   ESCAPE     :=  %1B
                   BS         :=  %08
                   LINDEL     :=  %7F
                   CR         :=  %0D
                   LF         :=  %0A
                   TXOFCH     :=  %13
                   TXONCH     :=  %11
                   INSIZ      :=  128          ! INTBUF SIZE !
                   OUTSIZ     :=  128          ! OUTBUF SIZE !
                   RBSIZ      :=  256          ! RING BUFFER SIZE !
                ! BIT POSITIONS IN MONITOR FLAG WORD !
                   TRPMDE     :=  0
                   ISTOP      :=  1
                   OSTOP      :=  2
                   SNDMDE     :=  3
                   LDMDE      :=  4
                   ESC        :=  5
                   TXMSK      :=  %6
```

```
                    COMDS     :=  11

                TYPE
                    SWITCH    RECORD [    FCW_      WORD
                                          PC        WORD ]


                INTERNAL
                $SECTION DATA_DEC
                $ABS 0
0000                INTBUF    ARRAY [128 BYTE]
0080                OUTBUF    ARRAY [128 BYTE]
0100                RNGBUF    ARRAY [256 BYTE]
0200                MCZBUF    ARRAY [256 BYTE]

0300                BUFADR    WORD
0302                BUFSIZ    WORD

0304                INTPTR    WORD
0306                OUTPTR    WORD

0308                UNIMP     WORD
030A                BRKCNT    WORD

030C                NXTPTR    WORD
030E                GETOUT    WORD
0310                MCZPUT    WORD
0312                MCZGET    WORD
0314                BRKSTR    WORD
0316                BRKADR    WORD
0318                TMPSP     WORD
031A                TMPFCW    WORD

031C                MFLAGS    WORD

                  ! USER REGISTER STORAGE !

031E                R0_       WORD
0320                R1_       WORD
0322                R2_       WORD
0324                R3_       WORD
0326                R4_       WORD
0328                R5_       WORD
032A                R6_       WORD
032C                R7_       WORD
032E                R8_       WORD
0330                R9_       WORD
0332                R10_      WORD
0334                R11_      WORD
0336                R12_      WORD
```

```
0338          R13_      WORD
033A          R14_      WORD
033C          R15_      WORD
033E          RPC_      WORD
0340          RFC_      WORD

0342          RETRY     WORD
0344          ADR_STR   WORD
0346          CMDTBL    ARRAY[12 WORD]

              INTERNAL
              $SECTION PSA_DATA
              $ABS 0
0000             PSA    RECORD [
                           DATA_AREA      WORD
                           CODE_AREA        WORD
                           UNIMP_INST     SWITCH
                           PRIV_INST      SWITCH
                           SYSTEM_CALL    SWITCH
                           SEG_TRAP       SWITCH
                           NMI_INT        SWITCH
                           NVI_INT        SWITCH
                           VEC_FCW        WORD
                           VEC_PC         ARRAY [200  WORD]
                           ]


              INTERNAL
              $SECTION BOOT_DATA
              $ABS 0
0000             BOOT_COM  RECORD [
                        TABLE_LOCK   WORD
                        SIGNAL       WORD
                        MSG1         WORD  ]

              $SECTION EXEC_PROC
              EXTERNAL
                 GETLNE     PROCEDURE
                 NMI        PROCEDURE
                 IMPINT     PROCEDURE
                 SNDCHR     PROCEDURE
                 SNDMSG     PROCEDURE
                 MEMINT     PROCEDURE
                 FAIL_SAFE  PROCEDURE
                 CLK_STORE  PROCEDURE
                 MCZEND     PROCEDURE
                 CONINT     PROCEDURE
                 DISPLAY    PROCEDURE
                 GETBUF     PROCEDURE
                 FILL       PROCEDURE
```

```
                    MOVE        PROCEDURE
                    REGISTER    PROCEDURE
                    GO          LABEL
                    BREAK       PROCEDURE
                    JUMP        PROCEDURE
                    NEXT        PROCEDURE
                    QUIT        PROCEDURE
                    SEND        PROCEDURE
                    LOADFL        PROCEDURE
                    GETCHR      PROCEDURE


                  GLOBAL EROR   LABEL
                  GLOBAL EXEC   LABEL
0000              GLOBAL INITL PROCEDURE
                  !*****************************************
                   *     DMONITOR INITIALIZATION           *
                   *****************************************!

                  ENTRY
                  ORGADR:
0000 F80F            JR         STARTP
                  PHYS_ID:
0002 F1F1            WVAL       %F1F1        ! UNIQUE PHYS_ID !
                  LOGO:
0004 0F              BVAL       %0F
0005 2A              BVAL       '*'
0006 5341            WVAL       'SA'
0008 5353            WVAL       'SS'
000A 2044            WVAL       ' D'
000C 4D4F            WVAL       'MO'
000E 4F49            WVAL       'NI'
0010 544F            WVAL       'TO'
0012 52              BVAL       'R'
0013 0D              BVAL       %0D

                  CMDCHR:
0014 4446            WVAL       'DF'
0016 4D52            WVAL       'MR'
0018 514C            WVAL       'QL'
001A 4A4E            WVAL       'JN'
001C 4753            WVAL       'GS'
001E 4220            WVAL       'B '
```

```
                ! START OF INITIAL ENTRY TO DMONITOR !
                STARTP:
                !*****************************************
                 *  RESET CODE AREA AND DATA AREA REGS   *
                 *****************************************!

                ! RESET DATA_AREA AND CODE_AREA REGISTERS !
0020 7D25          LDCTL     R2,PSAPOFF
0022 612E          LD        R14,DATA_AREA(R2)
0024 0000
0026 8DE4          TEST      R14
0028 EE02          JR        NZ,SET_OTHERS
002A 210E          LD        R14,#%7A00
002C 7A00
                SET_OTHERS:
002E 340C          LDAR      R12,ORGADR
0030 FFCE
0032 6F2C          LD        CODE_AREA(R2),R12  ! SAVE NEW CODE !
0034 0002
                                                ! BASE ADR !
0036 6F2E          LD        DATA_AREA(R2),R14
0038 0000

                !*****************************************
                 *                                       *
                 *  SFTW_INIT: INITIALIZES ALL BASIC     *
                 *             DATA STRUCTURES FOR THE   *
                 *             NORMAL FUNCTIONING OF     *
                 *             DMONITOR.                 *
                 *                                       *
                 *****************************************!

                ! CLEAR DMONITOR RAM AREA !
003A 76F2          LDA       R2,BUFADR(R14)    ! CLR DMONITOR RAM !
003C 0300
003E A121          LD        R1,R2
0040 A911          INC       R1,#2
0042 2103          LD        R3,#66
0044 0042
0046 0D25          LD        @R2,#0
0048 0000

004A BB21          LDIR      @R1,@R2,R3
004C 0310

                ! LD FIXED DATA IN RAM !
004E 34E7          LDA       R7,R14(#INTBUF)
0050 0000
0052 33E7          LD        R14(#INTPTR),R7
0054 0304
```

```
0056 34E7      LDA       R7,R14(#OUTBUF)
0058 0080
005A 33E7      LD        R14(#CUTPTR),R7
005C 0306
005E 34E3      LDA       R3,R14(#UNIMP)
0060 0308
0062 0D35      LD        @R3,#%0E00
0064 0E00
0066 34E3      LDA       R3,R14(#BRKCNT)
0068 030A

006A 4DE8      CLR       GETOUT(R14)
006C 030E
006E 4DE8      CLR       NXTPTR(R14)
0070 030C

               !*****************************************
                *   INITIALIZE PROGRAM STATUS AREA      *
                *****************************************!

0072 7D25      LDCTL     R2,PSAPOFF
0074 76C4      LDA       R4,FAIL_SAFE(R12)
0076 0000*
0078 A923      INC       R2,#4
007A 2F24      LD        @R2,R4
007C A123      LD        R3,R2
007E A931      INC       R3,#2      ! FAILSAFE PSA !
0080 2104      LD        R4,#24
0082 0018
0084 BB21      LDIR      @R3,@R2,R4   ! SETUP PSA !
0086 0430

0088 7D25      LDCTL     R2,PSAPOFF
008A 2101      LD        R1,#%4000
008C 4000
008E 3423      LDA       R3,R2(#UNIMP_INST)
0090 0004
0092 76C4      LDA       R4,IMPINT(R12)
0094 0000*
0096 3334      LD        R3(#2),R4
0098 0002

009A 8D48      CLR       R4
               DO
009C 7331         LD     R3(R4),R1
009E 0400
00A0 A943         INC    R4,#4
00A2 0B04         CP     R4,#28
00A4 001C
00A6 E601         JR     Z,LD_PC
```

```
00A8 E8F9       OD
              LD_PC:

00AA 3423       LDA       R3,P2(#NMI_INT)     ! LOAD NMI HDLR !
00AC 0014
00AE 76C4       LDA       R4,NMI(R12)
00B0 0000*
00B2 3334       LD        R3(#2),R4
00B4 0002

                      ! SET INTERRUPT HANDLERS !
00B6 3423       LDA         R3,R2(#VEC_PC) ! BASE OF INT VEC!
00B8 001E
00BA 34C2       LDA         R2,R12(#CONINT) !CONS INPUT!
00BC 0000*
00BE 3332       LD          R3(#0),R2
00C0 0000

                         ! MCZ INPUT !
00C2 34C2       LDA         R2,R12(#MCZHND)
00C4 0000*
00C6 3332       LD          R3(#4),P2
00C8 0004

                       ! MEMORY ACCESS VIOLATION !
00CA 34C2       LDA         R2,R12(#MEMINT)
00CC 0000*
00CE 3332       LD          R3(#14),R2
00D0 000E

              !*****************************************
               *            SET STACK POINTER          *
               *****************************************!

00D2 A1EF       LD        R15,R14
00D4 010F       ADD       P15,#%05F0   ! SET STACK POINTER !
00D6 05F0
00D8 6FEF       LD        R15_(R14),R15
00DA 033C

              !*****************************************
               *  INITIALIZE COMMAND JUMP TABLE        *
               *****************************************!

                    ! INIT COMMAND JUMP TABLE !
00DC 34E1       LDA       R1,R14(#CMDTBL)  ! BASE ADR !
00DE 0346
00E0 34C2       LDA       R2,R12(#DISPLAY)
00E2 0000*
00E4 3312       LD        R1(#0),R2
```

```
00E6 0000
00E8 34C2      LDA       R2,R12(#FILL)
00EA 0000*
00EC 3312      LD        R1(#2),R2
00EE 0002
00F0 34C2      LDA       R2,R12(#MOVE)
00F2 0000*
00F4 3312      LD        R1(#4),R2
00F6 0004
00F8 34C2      LDA       R2,R12(#REGISTER)
00FA 0000*
00FC 3312      LD        R1(#6),R2
00FE 0006
0100 34C2      LDA       R2,R12(#QUIT)
0102 0000*
0104 3312      LD        R1(#8),R2
0106 0008
0108 34C2      LDA       R2,R12(#LOADFL)
010A 0000*
010C 3312      LD        R1(#10),R2
010E 000A
0110 34C2      LDA       R2,R12(#JUMP)
0112 0000*
0114 3312      LD        R1(#12),R2
0116 000C
0118 34C2      LDA       R2,R12(#NEXT)
011A 0000*
011C 3312      LD        R1(#14),R2
011E 000E
0120 34C2      LDA       R2,R12(#GO)
0122 0000*
0124 3312      LD        R1(#16),R2
0126 0010
0128 34C2      LDA       R2,R12(#SEND)
012A 0000*
012C 3312      LD        R1(#18),R2
012E 0012
0130 34C2      LDA       R2,R12(#BREAK)
0132 0000*
0134 3312      LD        R1(#20),R2
0136 0014

          ! DISPLAY LOGO !
0138 3402      LDAR        R2,LOGO
013A FEC8
013C 34CA      LDA       R10,R12(#SNDMSG)
013E 0000*
0140 1FA0      CALL      @R10
```

```
                    ! R12 = START OF CODE !
                    ! R14 = START OF DATA AREA !
                    ! R15 = STACK POINTER !

                   !*****************************************
                    *      BASIC CONTROL OF DMONITOR        *
                    *****************************************!
                   EXEC:
0142 340C             LDAR      R12,ORGADR
0144 FEBA
0146 7D15             LDCTL     R1,PSAPOFF
0148 211E             LD        R14,@R1
014A 3401             LDAR      R1,EXEC
014C FFF4
014E 93F1             PUSH      @R15,R1
0150 31E1             LD        R1,R14(#MFLAGS)      ! CLR MON FLAGS !
0152 031C
0154 0701             AND       R1,#TXMSK      ! EXCEPT TX BIT !
0156 0006
0158 33E1             LD        R14(#MFLAGS),R1
015A 031C
015C 7C05             EI        VI

015E C82A             LDB       RL0,#'*'
0160 76CA             LDA       R10,SNDCHR(R12)    ! CALR SUBSTIT !
0162 0000*
0164 1FA0             CALL      @R10
0166 34CA             LDA       R10,R12(#GETLNE)   ! CALR SUBSTIT !
0168 0000*
016A 1FA0             CALL      @R10
                                                ! GET CMD 1ST CHR !
                   ! COMMAND LOOKUP ROUTINE !
016C 3401             LDAR      R1,CMDCHR       ! BASE OF LKUP TBLE!
016E FEA4
0170 BD2B             LDK       R2,#COMDS       ! TOTAL # CMDS !
0172 BA14             CPIRB     RL0,@R1,R2,EQ    ! FIND CMD !
0174 0286

0176 EE09             JR        NZ,EROR           ! CMD NOT FOUND !
0178 BD1A             LDK       R1,#COMDS - 1  ! DETERMINE INDEX !
017A 8321             SUB       R1,R2
017C B311             SLL       R1,#1
017E 0001
0180 34E2             LDA       R2,R14(#CMDTBL)
0182 0346
0184 8121             ADD       R1,R2
0186 2113             LD        R3,@R1          ! GET CMD ADR !
0188 1E38             JP        @R3             ! GOTO CMD HDLR !
```

```
                !*****************************************
                 *           GLOBAL ERROR LABEL          *
                 *****************************************!

                ERROR:
0188 C83F         LDB       RL0,#'?'       ! DISP '?' FOR ER !
018C 34CA         LDA       R10,R12(#SNDCHR)   ! CALR SUBSTIT !
018E 0000*
0190 1FA0         CALL      @R10
0192 A1EF         LD        R15,P14
0194 010F         ADD       R15,#%05F0     ! RESET STACK POINTER !
0196 05F0
0198 34E1         LDA       P1,R14(#OUTBUF)
019A 0080
019C 33E1         LD        R14(#OUTPTR),R1
019E 0306
01A0 E8D0         JR        EXEC           ! DELETE CMD !

01A2            END INITL

                END EXEC_DMONITOR
```

2. INTERRUPT MODULE

```
Z8003ASM  2.02
LOC     OBJ CODE     STMT SOURCE STATEMENT

                        1 INTERRUPT_HDLR MODULE
                $LISTON  $TTY

                !*****************************************
                 *                                       *
                 *      INTERRUPT HANDLER ROUTINES       *
                 *                                       *
                 *****************************************!

                CONSTANT

                   RXR       :=  2
                   TXR       :=  0
                   PAR       :=  7
                   PORTAD    :=  %FFD9
                   PORTBD    :=  %FFE1
                   PORTAC    :=  %FFDB
                   PORTBC    :=  %FFE3

                   IDPORT    :=  %FFCB
                   ICPORT    :=  %FFC9

                   TCMD    :=  %FFD2
                   TDTA    :=  %FFD0

                   BUS_LOCK  :=%FFF9
                   BUS_UNLOCK :=%FFF9
                   VINTR     :=  %(2)0001000000000000
                   VIBIT     :=  12
                   ESCAPE    :=  %1B
                   BS        :=  %08
                   LINDEL    :=  %7F
                   CR        :=  %0D
                   LF        :=  %0A
                   TXOFCH    :=  %13
                   TXONCH    :=  %11
                   INSIZ     :=  128        ! INTBUF SIZE !
                   OUTSIZ    :=  128        ! OUTBUF SIZE !
                   RBSIZ     :=  256        ! RING BUFFER SIZE !
                ! BIT POSITIONS IN MONITOR FLAG WORD !
                   TRPMDE    :=  0
                   ISTOP     :=  1
                   OSTOP     :=  2
                   SNDMDE    :=  3
```

```
                LDMDE     :=  4
                ESC       :=  5
                TXMSK     := %6


              EXTERNAL
                EXEC      LABEL
                PBUFNC    LABEL
                CONVW     PROCEDURE
                PRNTBF    PROCEDURE
                BPKROU    LABEL
                NEWLNE    PROCEDURE

              INTERNAL
              $SECTION DATA_DEC
              $ABS 0

0000            INTBUF    ARRAY [128 BYTE]
0080            OUTBUF    ARRAY [128 BYTE]
0100            RNGBUF    ARRAY [256 BYTE]
0200            MCZBUF    ARRAY [256 BYTE]

0300            BUFADR    WORD
0302            BUFSIZ    WORD

0304            INTPTR    WORD
0306            OUTPTR    WORD

0308            UNIMP     WORD
030A            BRKCNT    WORD

030C            NXTPTR    WORD
030E            GETOUT    WORD
0310            MCZPUT    WORD
0312            MCZGET    WORD
0314            BRKSTR    WORD
0316            BRKADR    WORD
0318            TMPSP     WORD
031A            TMPFCW    WORD

031C            MFLAGS    WORD

              ! USER REGISTER STORAGE !

031E            R0_       WORD
0320            R1_       WORD
0322            R2_       WORD
0324            R3_       WORD
0326            R4_       WORD
0328            R5_       WORD
```

```
032A            R6_       WORD
032C            R7_       WORD
032E            R8_       WORD
0330            R9_       WORD
0332            R10_      WORD
0334            R11_      WORD
0336            R12_      WORD
0338            R13_      WORD
033A            R14_      WORD
033C            R15_      WORD
033E            RPC_      WORD
0340            RFC_      WORD

0342            RETRY     WORD
0344            ADR_STR   WORD

              GLOBAL
              $SECTION INTERRUPT_PROC
              $REL 0

0000 0C  49     SECMSG ARRAY [* BYTE]:='%0CILL MEM REF%0D'
0002 4C  4C
0004 20  4D
0006 45  4D
0008 20  52
000A 45  46
000C 0D
000D 09  42     BRKMSG  ARRAY [*  BYTE]:='%09BREAK AT '
000F 52  45
0011 41  4B
0013 20  41
0015 54  20
0017 05  4E     NMIMSG  ARRAY [*  BYTE]:='%05NMI %0D'
0019 4D  49
001B 20  0D
001D 09  55     UNK_INT  ARRAY [* BYTE]:='%09UNK TRAP%0D'
001F 4E  4B
0021 20  54
0023 52  41
0025 50  0D
```

```
0028            SAVRG   PROCEDURE
                !*****************************************
                 *                                       *
                 *  SAVRG: SAVES USER PROGRAM STATUS     *
                 *         AND REGS 1-14 CONTENTS.       *
                 *                                       *
                 *****************************************!
                ENTRY
0028 6FEF          LD       TMPSP(R14),R15      ! RTN ADR !
002A 0318
002C 31F0          LD       R0,R15(#4)      !SAVE:  !
002E 0004
0030 6FE0          LD       RFC_(R14),R0         ! USER FCW !
0032 0340
0034 31F0          LD       R0,R15(#6)
0036 0006
0038 6FE0          LD       RPC_(R14),R0         ! USER PC !
003A 033E
                ! SAVE R1 - R14 !
003C 76EF          LDA      R15,R1_(R14)
003E 0320
0040 1CF9          LDM      @R15,R1,#11    ! STORE REGS !
0042 010A
0044 61EF          LD       R15,TMPSP(R14)      ! RESTORE SP !
0046 0318
0048 6FED          LD       R13_(R14),R13
004A 0338
004C 9E08          RET
004E            END SAVRG

                GLOBAL
004E            DMON_ENTRY  PROCEDURE
                !*****************************************
                 *                                       *
                 *  DMON_ENTRY: RESTORES DMONITOR R12    *
                 *                AND R14 (CODE AND DATA)*
                 *                REGISTERS FOR INTERRUPT*
                 *                ENTRIES.               *
                 *                                       *
                 *****************************************!
                ENTRY
004E 93F1          PUSH     @R15,R1
0050 93FE          PUSH     @R15,R14
0052 7D15          LDCTL    R1,PSAPOFF     ! GET PSA BASE !
0054 211E          LD       R14,@R1        ! RESTORE DATA BASE !
0056 6FEC          LD       R12_(R14),R12  ! SAVE USER R12 !
0058 0336
005A 311C          LD       R12,R1(#2)     ! RESTORE CODE BASE !
005C 0002
005E 97F1          POP      R1,@R15
```

```
0060 6FE1      LD       R14_(R14),R1
0062 033A
0064 97F1      POP      R1,@R15
0066 9E08      RET

0068        END DMON_ENTRY

            GLOBAL
0068        MEMINT  PROCEDURE
            !*****************************************
             *                                       *
             *  MEMINT: INTERRUPT HANDLER TO SIGNAL *
             *          USER OF ILLEGAL USER MODE   *
             *          USE OF LOCAL MEMORY, AND    *
             *          TERMINATE USER PROGRAM EXE- *
             *          CUTION WITH RETURN TO DMON. *
             *                                       *
             *****************************************!
            ENTRY
0068 D00E      CALR     DMON_ENTRY
006A 33EF      LD       R14(#R15_),R15
006C 033C
006E 33E0      LD       R14(#R0_),R0
0070 031E
0072 D026      CALR     SAVRG      ! SAVE USER REGS !
0074 3402      LDAR     R2,SECMSG
0076 FF88
0078 93F2      PUSH     @R15,R2    ! SAVE MSG ADR !
007A E84D      JR       ALERT
007C        END MEMINT

            GLOBAL
007C        IMPINT  PROCEDURE
            !*****************************************
             *                                       *
             *  IMPINT: NON-IMPLEMENTED INSTRUCTION *
             *          INTERRUPT HANDLER USED TO   *
             *          TERMINATE USER PROGRAM EXE- *
             *          CUTION AT A PRESET BREAK    *
             *          POINT. ALL USER REGISTERS   *
             *          AND PROGRAM STATUS IS SAVED *
             *                                       *
             *****************************************!
            ENTRY
007C D018      CALR     DMON_ENTRY
007E 33EF      LD       R14(#R15_),R15
0080 033C
0082 33E0      LD       R14(#R0_),R0
0084 031E
0086 D030      CALR     SAVRG          ! SAVE REGS !
```

```
0088 6BE1      DEC       RPC_(R14),#2
008A 033E
008C 6BE0      DEC       BRKCNT(R14),#1
008E 030A
0090 EE12      JR        NZ,SSTP       ! NO, EXEC 1 INST !
0092 4DE5      LD        BRKCNT(R14),#%0001
0094 030A
0096 0001

0098 7C05      EI        VI
009A 3402      LDAR        R2,BRKMSG    ! LOAD ADR BRK MSG !
009C FF6F
009E DFB0      CALR      SNDMSG
00A0 31E5      LD        R5,R14(#RPC_)
00A2 033E
00A4 34CA      LDA       R10,R12(#CONVW)
00A6 0000*
00A8 1FA0      CALL      @R10          ! CONVERT BYTE !
00AA 34CA      LDA       R10,R12(#PRNTBF)
00AC 0000*
00AE 1FA0      CALL      @R10          ! OUTPUT TO CONS !
00B0 34CA      LDA       R10,R12(#EXEC)
00B2 0000*
00B4 1EA8      JP        @R10          ! RETURN TO EXEC !

               SSTP:
00B6 34CA        LDA     R10,R12(#BRKROU)
00B8 0000*
00BA 1EA8        JP      @R10          ! JUMP TO BRK, !
                                       ! ROUTINE !
00BC           END IMPINT

               GLOBAL
00BC           FAIL_SAFE PROCEDURE
               !*****************************************
                *                                       *
                *  FAIL_SAFE: RESTORES PROGRAM TO       *
                *             CONTROL OF DMONITOR ON AN *
                *             UNKNOWN INTERRUPT OR TRAP *
                *             SOURCE.                   *
                *       RETURNS TO THE BASIC EXEC.      *
                *                                       *
                *****************************************!
               ENTRY
00BC D038        CALR      DMON_ENTRY
00BE 33EF        LD        R14(#R15_),R15
00C0 033C
00C2 33E0        LD        R14(#R0_),R0
00C4 031E
00C6 D050        CALR      SAVRG       ! SAVE USER REGS !
```

```
00C8 3402       LDAR    R2,UNK_INT
00CA FF51
00CC 93F2       PUSH    @R15,R2     ! SAVE MSG ADR !
00CE F823       JR      ALERT
00D0          END FAIL_SAFE

00D0          NMI  PROCEDURE
              !*****************************************
               *                                       *
               *  NMI INT: NON-MASKABLE INTERRUPT HDLR *
               *           IN RESPONSE TO MULTIPLE     *
               *           INT SOURCES. IF IN TRANS-   *
               *    PARENT MODE, IT WILL SEND THE      *
               *    ESCAPE CHAR TO MCZ AND RESTORE     *
               *    THE STACK; IF EXECUTING USER       *
               *    PROGRAM, IT WILL SAVE PROGRAM      *
               *    STATUS AND ALL REGISTERS.          *
               *                                       *
               *****************************************!
              ENTRY
00D0 93FC       PUSH    @R15,R12
00D2 93FE       PUSH    @R15,R14
00D4 93F1       PUSH    @R15,R1
00D6 7D15       LDCTL   R1,PSAPOFF
00D8 211E       LD      R14,@R1     ! LOAD DATA_AREA BASE !
00DA 311C       LD      R12,R1(#2)  ! LOAD CODE_AREA BASE !
00DC 0002
00DE 97F1       POP     R1,@R15     ! RESTORE R1 !

00E0 67E0       BIT     MFLAGS(R14),#TRPMDE
00E2 031C
00E4 EE0C       JR      NZ,TQUIT        ! YES, TERMINATE !

              ! EXECUTING USER PROGRAM !
00E6 97FE       POP     R14,@R15
00E8 97FC       POP     R12,@R15
00EA D04F       CALR    DMON_ENTRY
00EC 33EF       LD      R14(#R15_),R15
00EE 033C
00F0 33E0       LD      R14(#R0_),R0
00F2 031E

00F4 D067       CALR    SAVRG           ! SAVE PS AND REGS !
00F6 3402       LDAR      R2,NMIMSG     ! LOAD ALERT MSG !
00F8 FF1D
00FA 93F2       PUSH     @R15,R2
00FC E80C       JR      ALERT

              ! IN TRANSPARENT MODE !
              TQUIT:
```

```
00FE 63E0      RES       MFLAGS(R14),#TRPMDE
0100 031C
0102 A1EF      LD        R15,R14
0104 010F      ADD       R15,#%05F0     ! SET STACK PTR !
0106 05F0
0108 6FEF      LD        R15_(R14),R15 ! SET USER SP !
010A 033C
010C C81B      LDB       RL0,#ESCAPE
010E DFF1      CALR      SNDMCZ    ! SEND ESCAPE TO MCZ !
0110 3402      LDAR        R2,NMIMSG     ! LOAD ALERT MSG !
0112 FF03
0114 93F2      PUSH       @R15,R2

               !OUTPUT NMI  ALERT MSG AND RETURN TO EXEC !
               ALERT:
0116 7C05      EI        VI
0118 76CA      LDA       R10,NEWLNE(R12)
011A 0000*
011C 1FA0      CALL      @R10

011E 97F2      POP       R2,@R15
0120 DFF1      CALR      SNDMSG
0122 A1EF      LD        R15,R14
0124 010F      ADD       R15,#%05F0    ! RESET STACK POINTER !
0126 05F0
0128 34CA      LDA       R10,R12(#EXEC)
012A 0000*
012C 1EA8      JP        @R10        ! RETURN TO DMONITOR !
012F           END NMI

               GLOBAL
012E           SNDMCZ  PROCEDURE
               !*****************************************
                *                                       *
                *  SNDMCZ: OUTPUT CHAR TO SERIAL PORT   *
                *          ONE (MCZ SYS).               *
                *                                       *
                *  REG USE: INPUT  RL0 = CHAR           *
                *           RETURN Z IF CHAR = CR       *
                *                                       *
                *****************************************!
               ENTRY
012E 3A04      INB       RH0,PORTAC     ! GET STATUS !
0130 FFDB
0132 A600      BITB      RH0,#TXR        ! TRANSMIT RDY? !
0134 E6FC      JR        Z,SNDMCZ       ! NOT YET..... !
0136 3A86      OUTB      PORTAD,RL0     ! YES, SND CHR !
0138 FFD9
013A 0A08      CPB       RL0,#CR
013C 0D0D
```

```
013E 9E08       RET
0142            END SNDMCZ

                GLOBAL
0140            SNDMSG  PROCEDURE
                !*****************************************
                 *                                       *
                 *  SNDMSG: SEND MSG SPECIFIED TO CONS   *
                 *          (PORT2). FIRST BYTE OF MSG   *
                 *          IS THE DECIMAL LENGTH IN     *
                 *          WORDS.                       *
                 *                                       *
                 *  REG USE: INPUT  R2 = MSG ADDR        *
                 *                                       *
                 *****************************************!
                ENTRY
0140 34E1       LDA     R1,R14(#OUTBUF)
0142 0080
0144 8D08       CLR     R0
0146 2028       LDB     RL0,@R2      ! GET BYTE COUNT !
0148 8101       ADD     R1,R0
014A 33E1       LD      R14(#OUTPTR),R1
014C 0306
014E A920       INC     R2       ! SETUP FOR TRANSFER !
0150 34E1       LDA      R1,R14(#OUTBUF)
0152 0080
0154 BA21       LDIRB    @R1,@R2,R0  ! TRANSFER TO OUTBUF !
0156 0010
0158 34CA       LDA     R10,R12(#PBUFNC)
015A 0000*
015C 1EA8       JP      @R10       ! OUTPUT TO CONS !
015E            END SNDMSG

                GLOBAL
015E            CONINT  PROCEDURE
                !*****************************************
                 *                                       *
                 *  CONINT: CONS (PORT2) INPUT INT HDLR  *
                 *          ROUTINE, WHICH GETS REC CHR  *
                 *          FROM USART. REC CHR IS CK    *
                 *          FOR TXOFCH OR TXONCH, AND    *
                 *          MFLAGS ADJUSTED ACCORDINGLY  *
                 *          TO SIGNAL PROCEDURES.        *
                 *                                       *
                 *****************************************!
                ENTRY
015E 93F0       PUSH    @R15,R0
0160 93F1       PUSH    @R15,R1
0162 93F2       PUSH    @R15,R2      ! SAVE WORK REGS !
0164 93FE       PUSH    @R15,R14
```

```
0166 7D15      LDCTL    R1,PSAPOFF  ! DATA_AREA ADR !
0168 211E      LD       R14,@R1

               ! GET CHAR AND CHECK FOR TXOFCH OR TXONCH !
016A 3A94      INB      RL1,PORTBD  ! GET USART DATA !
016C FFE1
016E A297      RESB     RL1,#PAR    ! CLR PARITY BIT !
0170 67E0      BIT      MFLAGS(R14),#TRPMDE
0172 031C
                                    ! TRANSPARENT MODE !
0174 EE18      JR       NZ,PUTCHR   ! YES,.....!
0176 0A09      CPB      RL1,#TXONCH   ! NO, CK FOR TXONCH!
0178 1111
017A EE03      JR       NZ,AGAIN    ! NO,....!
017C 63E2      RES      MFLAGS(R14),#OSTOP  ! RESET TO, !
017E 031C
                                      ! RESUME OUTPUT !
0180 E818      JR       FINISH
               AGAIN:
0182 0A09      CPB      RL1,#TXOFCH     ! CK FOR TXOFCH !
0184 1313
0186 EE03      JR       NZ,AGAIN2       ! NO,.... !
0188 65E2      SET      MFLAGS(R14),#OSTOP  ! STOP OUTPUT !
018A 031C
018C E812      JR       FINISH

               ! CHECK FOR ESCAPE CHARACTER !
               AGAIN2:
018E 0A09      CPB      RL1,#ESCAPE
0190 1B1B
0192 EE09      JR       NZ,PUTCHR   ! NO,.........!
0194 67E3      BIT      MFLAGS(R14),#SNDMDE ! YES, CK SND MDE!
0196 031C
0198 EE03      JR       NZ,ESCP     ! YES,.........!
019A 67E4      BIT      MFLAGS(R14),#LDMDE ! NO, CK LD MDE !
019C 031C
019E E603      JR       Z,PUTCHR    ! NO !
               ESCP:
01A0 65E5      SET      MFLAGS(R14),#ESC  ! SET ESCAPE BIT !
01A2 031C
01A4 E806      JR       FINISH

               ! PRIMARY SAVE CHARACTER ROUTINE !
               PUTCHR:
01A6 31E0      LD       R0,R14(#NXTPTR)
01A8 030C
01AA DFF8      CALR     GETBUF      ! GET RNGBUF ADDR !
01AC 33E0      LD       R14(#NXTPTR),R0
01AE 030C
01B0 2E29      LD3      @R2,RL1     ! PUT CHR IN RNGBUF !
```

```
                FINISH:
01B2 97FE          POP      R14,@R15
01B4 97F2          POP      R2,@R15
01B6 97F1          POP      R1,@R15
01B8 97F0          POP      R0,@R15     ! RESTORE WORK REGS !
01BA 7B00          IRET
01BC            END CONINT

                GLOBAL
01BC            GETBUF  PROCEDURE
                !*****************************************
                 *                                       *
                 *  GETBUF: DETERMINES POSITION IN       *
                 *          RINGBUFFER TO PUT OR GET     *
                 *          NEXT CHAR.                   *
                 *                                       *
                 *  REG USE: INPUT  R0 = CURRENT INDEX   *
                 *           RETURN R0 = NEW INDEX       *
                 *                  R2 = ADR OF RNGBUF   *
                 *                                       *
                 *****************************************!
                ENTRY
01BC 93FD          PUSH     @R15,R13
01BE A102          LD       R2,R0
01C0 A900          INC      R0,#1          ! INC PTR !
01C2 0B00          CP       R0,#RBSIZ      ! WRAP AROUND !
01C4 0100
01C6 EE01          JR       NZ,GB
01C8 8D08          CLR      R0             ! RESET INDEX !
01CA 34ED       GB:LDA      R13,R14(#RNGBUF)   ! NEW ADR !
01CC 0100
01CE 81D2          ADD      R2,R13
01D0 97FD          POP      R13,@R15
01D2 9E08          RET
01D4            END GETBUF
```

```
                GLOBAL
01D4            MCZHND  PROCEDURE
                !*****************************************
                 *                                       *
                 *  MCZHND: MCZ (SERIAL PORT1) INPUT     *
                 *          INTERRUPT HANDLER ROUTINE    *
                 *          WHICH GETS RECEIVED CHAR     *
                 *          FROM USART, AND STORES IN    *
                 *          MCZBUF.                      *
                 *                                       *
                 *****************************************!
                ENTRY
01D4 93F0           PUSH    @R15,R0
01D6 93F1           PUSH    @R15,R1
01D8 93F2           PUSH    @R15,R2      ! SAVE WORK REGS !
01DA 93FE           PUSH    @R15,R14
01DC 7D15           LDCTL   R1,PSAPOFF
01DE 211E           LD      R14,@R1     ! DATA_AREA ADR !
                ! GET CHAR FROM MCZ !
01E0 3A94           INB     RL1,PORTAD   ! GET CHR !
01E2 FFD9
01E4 A297           RESB    RL1,#PAR     ! RESET PARITY !
01E6 31E0           LD      R0,R14(#MCZPUT)
01E8 0310
01EA DFF8           CALR    GMCZAD       ! GET MCZBUF ADR !
01EC 33E0           LD      R14(#MCZPUT),R0
01EE 0310
01F0 2E29           LDB     @R2,RL1     ! SAVE CHAR !

                !RESTORE WORK REGS !
01F2 97FE           POP     R14,@R15
01F4 97F2           POP     R2,@R15
01F6 97F1           POP     R1,@R15
01F8 97F0           POP     R0,@R15
01FA 7B00           IRET
01FC            END MCZHND
```

```
                GLOBAL
01FC            GMCZAD  PROCEDURE
                !*****************************************
                 *                                       *
                 *  GMCZAD: GET NEXT ADR OF MCZ BUFFER   *
                 *          TO STORE OR GET CHARACTER.   *
                 *                                       *
                 *  REG USE: INPUT  R0 = PTR IN MCZBUF   *
                 *           RETURN R0 = NEW PTR IN BUF  *
                 *                  R2 = BGN OF MCZBUF   *
                 *                                       *
                 *****************************************!
                ENTRY
01FC 93FD          PUSH     @R15,R13
01FE A102          LD       R2,R0
0200 A900          INC      R0,#1
0202 0B00          CP       R0,#RBSIZ      ! WRAP AROUND? !
0204 0100
0206 EE01          JR       NZ,GBZ
0208 8D08          CLR      R0             ! RESET OFFSET !
                GBZ:
020A 34ED          LDA      R13,R14(#MCZBUF)    ! GET ADR !
020C 0200
020E 81D2          ADD      R2,R13
0210 97FD          POP      R13,@R15
0212 9E08          RET
0214            END GMCZAD

                END INTERRUPT_HDLR
```

3. LOAD MODULE

```
Z8000ASM  2.02
LOC    OBJ CODE     STMT SOURCE STATEMENT

                        1 LOAD_CMD MODULE
                $LISTON  $TTY

                !****************************************
                 *                                      *
                 *  LOAD CMD: COMMAND TO DOWNLOAD Z8K   *
                 *            OBJ CODE FILE FROM RIO/   *
                 *  MCZ SYSTEM IN TEXTRONIX FORMATTED   *
                 *  PACKET PASSING PROTOCOL. USER CAN   *
                 *  SPECIFY LOAD ADDRESS INSTEAD OF     *
                 *  USING MCZ LOAD ADDRESS.             *
                 *                                      *
                 *           CAUTION:                   *
                 *    CODE FILES MUST BE RELOCATABLE    *
                 *    TO EXECUTE PROPERLY.              *
                 *                                      *
                 *     SYNTAX:  LOAD <FILENAME> [<ADR>] *
                 *                                      *
                 ****************************************!

                CONSTANT

                   RXR        :=  2
                   TXR        :=  0
                   PAR        :=  7
                   PORTAD     :=  %FFD9
                   PORTBD     :=  %FFE1
                   PORTAC     :=  %FFDB
                   PORTBC     :=  %FFE3

                   IDPORT     :=  %FFCB
                   ICPORT     :=  %FFC9

                   TCMD    :=  %FFD2
                   TDTA    :=  %FFD0

                   BUS_LOCK :=%FFF9
                   BUS_UNLOCK :=%FFF8
                   VINTR      :=  %(2)0001000000000000
                   VIBIT      :=  12
                   ESCAPE     :=  %1B
                   BS         :=  %08
                   LINDEL     :=  %7F
                   CR         :=  %0D
```

```
                    LF          :=  %0A
                    TXOFCH      :=  %13
                    TXONCH      :=  %11
                    INSIZ       :=  128         ! INTBUF SIZE !
                    OUTSIZ      :=  128         ! OUTBUF SIZE !
                    RBSIZ       :=  256         ! RING BUFFER SIZE !
                  ! BIT POSITIONS IN MONITOR FLAG WORD !
                    TRPMDE      :=  0
                    ISTOP       :=  1
                    OSTOP       :=  2
                    SNDMDE      :=  3
                    LDMDE       :=  4
                    ESC         :=  5
                    TXMSK       := %6

                    COMDS       :=  11

                  INTERNAL
                  $SECTION DATA_DEC
                  $ABS 0
0000                INTBUF      ARRAY [128 BYTE]
0080                OUTBUF      ARRAY [128 BYTE]
0100                RNGBUF      ARRAY [256 BYTE]
0200                MCZBUF      ARRAY [256 BYTE]

0300                BUFADR      WORD
0302                BUFSIZ      WORD

0304                INTPTR      WORD
0306                OUTPTR      WORD

0308                UNIMP       WORD
030A                BRKCNT      WORD

030C                NXTPTR      WORD
030E                GETOUT      WORD
0310                MCZPUT      WORD
0312                MCZGET      WORD
0314                BRKSTR      WORD
0316                BPKADR      WORD
0318                TMPSP       WORD
031A                TMPFCW      WORD

031C                MFLAGS      WORD

                  ! USER REGISTER STORAGE !

031E                R0_         WORD
0320                R1_         WORD
0322                R2_         WORD
```

```
0324            R3_       WORD
0326            R4_       WORD
0328            R5_       WORD
032A            R6_       WORD
032C            R7_       WORD
032E            R8_       WORD
0330            R9_       WORD
0332            R10_      WORD
0334            R11_      WORD
0336            R12_      WORD
0338            R13_      WORD
033A            R14_      WORD
033C            R15_      WORD
033E            RPC_      WORD
0340            PFC_      WORD

0342            RETRY     WORD
0344            ADR_STR   WORD
0346            CMDTBL    ARRAY[12 WORD]


                EXTERNAL  PRNTBF        PROCEDURE
                EXTERNAL  GETNXT        PROCEDURE
                EXTERNAL  EROR          LABEL
                EXTERNAL  SNDCHR        PROCEDURE
                EXTERNAL  GETADR        PROCEDURE
                EXTERNAL  GMCZAD        PROCEDURE
                EXTERNAL  SNDMCZ        PROCEDURE
                EXTERNAL  CONVERT       PROCEDURE
                EXTERNAL  PBUFNC        LABEL
                EXTERNAL  SNDMSG        PROCEDURE
                EXTERNAL  CONVW         PROCEDURE


                $SECTION LOAD_PROC
                $REL 0


                GLOBAL
0000            FNAME PROCEDURE
                !*****************************************
                 *                                       *
                 *   FNAME: RESETS TWO PTRS TO MCZBUF    *
                 *          AND CHECKS FOR FILENAME.     *
                 *                                       *
                 *****************************************!
                ENTRY
0000 4DE8          CLR      MCZGET(R14)
0002 0312
0004 4DE8          CLR      MCZPUT(R14)        ! RESET BUFFER !
```

```
0006 0310
0008 34CA       LDA       R10,R12(#GETNXT)
000A 0000*
000C 1FA0       CALL      @R10            ! SKIP CMD ARG !
000E 0A08       CPB       RL0,#'A'
0010 4141
0012 E711       JR        C,DUN
0014 0A08       CPB       RL0,#'Z'+1
0016 5B5B
0018 EF0E       JR        NC,DUN          ! 1ST CHR IN (A..Z) !
001A 76CA       LDA       R10,GETNXT(R12)
001C 0000*
001E 1FA0       CALL      @R10            ! SKIP TO NEXT ARG !
0020 E607       JR        Z,NO_ADR        ! NO NEXT ARG !
0022 76CA       LDA       R10,GETADR(R12)
0024 0000*
0026 1FA0       CALL      @R10            ! GET USER SPECIFIED !
                                          ! ADDRESS !
0028 A13B       LD        R11,R3          ! SAVE USER ADR !
002A 6FE3       LD        ADR_STR(R14),R3
002C 0344
002E 9E08       RET

           NO_ADR:
0030 210B       LD        R11,#%FFFE      ! SIGNAL TO USE MCZ !
0032 FFFE
                                          ! ADDRESS !
0034 9E08       RET

           DUN:
0036 8D98       CLR       R9
0038 34CA       LDA       R10,R12(#ERCR)
003A 0000*
003C 1EA8       JP        @R10            ! ERROR, RTN TO EXEC !
003E       END FNAME
```

```
                GLOBAL
003E            CMDPAS PROCEDURE
                !*****************************************
                 *                                       *
                 *  CMDPAS: LOAD CMD PASSING MECHANISM   *
                 *          SENDS 'B;' PLUS CONS CMD     *
                 *  LINE TO MCZ AND CKS RESPONSES FOR    *
                 *  GOOD Z80 PROGRAM LOAD.               *
                 *                                       *
                 *  REG USE:  RETURN   NZ IF Z80 LOADED  *
                 *                     Z  IF NOT         *
                 *                                       *
                 *****************************************!
                ENTRY
003E 67F5          BIT      MFLAGS(R14),#ESC  ! CK FOR ESCAPE !
0040 031C
0042 E602          JR       Z,GCMD
0044 8D41          SETFLG   Z
0046 9E08          RET

                GCMD:
0048 C242          LDB      RH2,#'B'
004A CA3B          LDB      RL2,#';'
004C 6FE2          LD       OUTBUF(R14),R2     ! LOAD INIT 'B;' !
004E 0080
                                              ! FOR BRIEF MODE !
0050 76E2          LDA       R2,OUTBUF(R14)
0052 0080
0054 A921          INC      R2,#2
0056 76E1          LDA       R1,INTBUF(R14)
0058 0000
005A 2100          LD       R0,#%40          ! LD CMD IN OUTBUF !
005C 0040
005E BB11          LDIR     @R2,@R1,R0
0060 0020

0062 76E1          LDA      R1,OUTBUF(R14)
0064 0080
0066 0101          ADD      R1,#%80
0068 0080
006A 6FE1          LD       OUTPTR(R14),R1
006C 0306

006E DFB6          CALR     OUTSTM           ! OUTPUT BUFFER !
0070 DFE7          CALR     SKIPLN           ! SKIP MCZ ECHO !
0072 DFD4          CALR     MCZCOM           ! WAIT RESPONSE !
0074 0A09          CPB      RL1,#'B'
0076 4242
0078 EE02          JR       NZ,LDSTAT
007A DFEC          CALR     SKIPLN         ! SKIP MCZ ECHO !
```

```
007C DFD9     CALR    MCZCOM        ! WAIT RESPONSE !

              ! VERIFY LOAD STATUS !
              LDSTAT:
007E 0A09     CPB     RL1,#'9'      ! TEST LEGAL !
0080 3939
0082 E60E     JR      Z,RECACK      ! ACKNOWLEDGEMENTS !
0084 0A09     CPB     RL1,#'0'
0086 3030
0088 E60B     JR      Z,RECACK      ! REC GOOD ACK !
008A 0A09     CPB     RL1,#'7'
008C 3737
008E E608     JR      Z,RECACK

              ! NO ACKNOWLEDGEMENTS RECEIVED !
              ERMSG:
0090 DFD9     CALR    RECMSG        ! GET MCZ MSG !
0092 34CA     LDA     R10,R12(#SNDCHR)
0094 0000*
0096 1FA0     CALL    @R10          ! SEND TO CONS !
0098 0A08     CPB     RL0,#LF
009A 0A0A
009C 9E06     RET     Z             ! DONE !
009E E8F8     JR      ERMSG

              ! ACKNOWLEDGE RECEIVED !
              RECACK:
00A0 8D43     RESFLG  Z             ! RETURN NZ !
00A2 9E08     RET
00A4          END CMDPAS

              GLOBAL
              SKPB    LABEL
00A4          SKIPLN  PROCEDURE
              !*****************************************
               *                                       *
               *  SKIPLN: SKIP RECEIVED LINE FROM      *
               *          MCZ; RETURN FIRST CHAR OF    *
               *          NEXT LINE.                   *
               *                                       *
               *  REG USE:  RETURN   RL1 = 1ST CHR     *
               *                     AND NZ IF ESC     *
               *                                       *
               *****************************************!
              ENTRY
00A4 DFE3     CALR    RECMSG        ! SKIP OVER LINE !
00A6 0A08     CPB     RL0,#CR       ! THRU CR,LF !
00A8 0D0D
00AA EEFC     JR      NZ,SKIPLN
```

```
                SKPB:
00AC 2101         LD        R1,#%3000      ! DELAY FACTOR !
00AE 3000
                ! MAIN LOOP FOR RECEIVING CHAR !
                LOOP1:
00B0 61E0         LD        R0,MCZGET(R14)
00B2 0312
00B4 4BE0         CP        R0,MCZPUT(R14)     ! TEST FOR REC CHR !
00B6 0310
00B8 EE03         JR        NZ,RECHR       ! YES,..... !
00BA AB10         DEC       R1,#1          ! NO, WAIT AWHILE !
00BC EEF9         JR        NZ,LOOP1
00BE 9F06         RET       Z              ! FORCED EOL !

                RECHR:
00C0 DFFB         CALR      MCZCOM
00C2 0A09         CPB       RL1,#' '       ! CK 1ST=PRNT CHR !
00C4 2020
00C6 9E0D         RET       PL
00C8 DFF5         CALR      RECMSG
00CA E8F0         JR        SKPB
00CC            END SKIPLN

00CC            MCZCOM PROCEDURE
                !*****************************************
                 *                                       *
                 *  MCZCOM: LOOPS WAITING FOR RECEIVE    *
                 *          CHAR FROM MCZ BY SEEING IF   *
                 *          MCZBUF GETS CHAR. DOES       *
                 *          ADVANCE POINTER.             *
                 *                                       *
                 *  REG USE: RETURN  RL1 = CHR           *
                 *                                       *
                 *****************************************!
                ENTRY
00CC 61E0         LD        R0,MCZGET(R14)     ! CHECK MCZBUF !
00CE 0312
00D0 4BE0         CP        R0,MCZPUT(R14)     ! POINTERS !
00D2 0310
00D4 E6FB         JR        Z,MCZCOM       ! WAIT.....!
00D6 34CA         LDA       R10,R12(#GMCZAD)
00D8 0000*
00DA 1FA0         CALL      @R10           ! GET CHAR FROM BUF !
00DC 2029         LDB       RL1,@R2
00DE 9E08         RET
00E0            END MCZCOM
```

```
00EC          RECMSG PROCEDURE
              !*****************************************
               *                                       *
               *  RECMSG: LOOPS WAITING FOR REC CHR    *
               *          FROM MCZ. GETS CHAR AND      *
               *          DO NOT ADVANCE BUF PTR.      *
               *                                       *
               *  REG USE:  RETURNS   RL0 = CHR        *
               *                                       *
               *****************************************!
              ENTRY
00F0 61E0        LD        R0,MCZGET(R14)
00F2 0312
00F4 4BE0        CP        R0,MCZPUT(R14)    ! CK FOR REC !
00F6 0310
00F8 E6FB        JR        Z, RECMSG     ! WAIT..... !
00FA 34CA        LDA       R10,R12(#GMCZAD)
00FC 0000*
00FF 1FA0        CALL      @R10          ! GET 1ST CHAR !
00F0 6FE0        LD        MCZGET(R14),R0    ! RESTORE PTR !
00F2 0312
00F4 2028        LDB       RL0,@R2       ! RTN CHAR !
00F6 9E08        RET
00F8          END RECMSG

              GLOBAL
              OUTSTM    LABEL
00F8          OUTLNE    PROCEDURE
              !*****************************************
               *                                       *
               *  OUTLNE: OUTPUTS A LINE OF CHAR FROM  *
               *          OUTBUF TO MCZ WITH CR AT     *
               *          END.                         *
               *                                       *
               *  OUTSTM: OUTPUTS A LINE OF CHAR W/CR  *
               *                                       *
               *****************************************!
              ENTRY
00F8 61E2        LD        R2,OUTPTR(R14)
00FA 0306
00FC 0C25        LDB       @R2,#CR         !STORE CR IN BUF !
00FE 0D0D
0100 69E0        INC       OUTPTR(R14),#1      ! INC PTR !
0102 0306

              ! NO CR ENTRY POINT !
              OUTSTM:
0104 76E1        LDA       R1,OUTBUF(R14)
0106 0080

              ! MAIN LOOP !
```

```
                OVRAGN:
0108 2019           LDB     RL0,@R1
010A A910           INC     R1
010C 34CA           LDA     R10,R12(#SNDMCZ)
010E 0020*
0110 1FA0           CALL    @R10            ! SND CHR TO MCZ !
0112 E603           JR      Z,FINIS
0114 4BE1           CP      R1,OUTPTR(R14)
0116 0306
0118 E7F7           JR      C,OVRAGN        ! CK IF BUF EMPTY !

                ! FINISHED, RESET OUTPTR(R14) AND BLANK OUTBUF !
                FINIS:
011A 76E2           LDA     R2,OUTBUF(R14) ! RESET POINTER !
011C 0080
011E 6FE2           LD      OUTPTR(R14),R2
0120 0306
0122 2100           LD      R0,#OUTSIZ/2
0124 0040
0126 AB00           DEC     R0,#1           ! SET COUNT !
0128 4DE5           LD      OUTBUF(R14),#'  '    ! LOAD COUNT !
012A 0080
012C 2020
012E 76E2           LDA     R2,OUTBUF(R14)
0130 0080
0132 A121           LD      R1,R2
0134 A911           INC     R1,#2
0136 BB21           LDIR    @R1,@R2,R0      ! CLR BUFFER !
0138 0010
013A 9E08           RET
013C            END OUTLNE
```

```
                ABORTM    LABEL
                GODPAK    LABEL
0130            BADPAK    PROCEDURE
                !*****************************************
                 *                                       *
                 *  BADPAK: SENDS RESEND SIGNAL ('7')    *
                 *          TO MCZ FOR BAD CKSUM OR REC  *
                 *          NON-ASCII CHR.               *
                 *                                       *
                 *  ABORTM: SENDS ABORT SIGNAL ('9')     *
                 *          WHEN USER SELECTED.          *
                 *                                       *
                 *  GODPAK: SENDS ACK SIGNAL ('0') FOR   *
                 *          RECEIPT OF GOOD PACKET.      *
                 *                                       *
                 *****************************************!
                ENTRY
0130 C837          LDB       RL0,#'7'        ! LD RESEND SIG !
013E F803          JR        OUTALL
                ABORTM:
0140 C839          LDB       RL0,#'9'        ! LD ABORT SIG !
0142 F801          JR        OUTALL
                GODPAK:
0144 C830          LDB       RL0,#'0'        ! LD REC OK SIG !
                OUTALL:
0146 6EE8          LDB       OUTBUF(R14),RL0
0148 0090
014A 76ED          LDA       R13,OUTBUF(R14)
014C 0080
014E A9D0          INC       R13,#1
0150 6FFD          LD        OUTPTR(R14),R13
0152 0306
0154 D02F          CALR      OUTLNE          ! SEND MCZ SYSTEM !
0156 D05A          CALR      SKIPLN          ! SKIP ECHO !
0158 9E08          RET
015A            END BADPAK
```

```
                GLOBAL
015A            GETACK   PROCEDURE
                !*****************************************
                 *                                       *
                 *  GETACK: RECEIVE AND INTERPRET ACK    *
                 *          FROM MCZ. GOOD ACK = '2'     *
                 *                    BAD  ACK = '7'     *
                 *                    ABORT    = '9'     *
                 *                                       *
                 *  REG USE:  RETURN  Z,NC IF GOOD ACK   *
                 *                    NZ,NC IF BAD ACK   *
                 *                    NZ,C IF ABORT      *
                 *                                       *
                 *****************************************!
                ENTRY
015A D048          CALR     MCZCOM          ! GET CHR !
015C 0A09          CPB      RL1,#'0'        ! CK FOR ACK !
015E 3030
0160 EE04          JR       NZ,NACK         ! NO........ !
0162 D060          CALR     SKIPLN          ! YES, REC ACK !
0164 8D41          SETFLG   Z
0166 8D83          RESFLG   C
0168 9E08          RET

                ! CK FOR '7' AND '9' NON-ACKNOWLEDGEMENTS !
                NACK:
016A 0A09          CPB      RL1,#'7'        ! CK FOR RESEND !
016C 3737
016E EE04          JR       NZ,ABRT         ! NO.... !
0170 D067          CALR     SKIPLN
0172 8D43          RESFLG   Z
0174 8D83          RESFLG   C
0176 9E08          RET

                ! CHECK FOR ABORT !
                ABRT:
0178 0A09          CPB      RL1,#'9'
017A 3939
017C E602          JR       Z,ENDIT         ! YES, ABORT... !
017E D050          CALR     RECMSG          ! GET ANOTHER CHR !
0180 E8EC          JR       GETACK          ! TRY AGAIN.... !

                ENDIT:
0182 D070          CALR     SKIPLN
0184 8D43          RESFLG   Z
0186 8D81          SETFLG   C
0188 9E08          RET
018A            END GETACK
```

```
018A                LINRCT PROCEDURE
                    !*****************************************
                     *                                       *
                     *  LINRCT: RECEIVES LINE OF CHAR FROM   *
                     *          MCZ AFTER RECEIPT OF '/',    *
                     *          AND STORES IN INTBUF, ADDING*
                     *          CR AT END AND FILTERING OUT *
                     *          CONTROL CHARACTERS. (<20H)   *
                     *          (TRUNCATES AFTER 80 CHAR)    *
                     *                                       *
                     *****************************************!
                    ENTRY
                    ! WAIT FOR ASCII / !
018A D056              CALR     RECMSG
018C 0A08              CPB      RL0,#'/'
018E 2F2F
0190 EEFC              JR       NZ,LINRCT     ! WAIT !
                    ! BEGIN STORING CHARACTERS !
0192 76E4              LDA       R4,INTBUF(R14)
0194 0000
0196 CB50              LDB      RL3,#80       !SET LINE LENGTH !
                    ! STORE CHAR IN INTBUF !
                    LOPSTR:
0198 D05D              CALR     RECMSG         ! GET CHAR !
019A 2E48              LDB      @R4,RL0        ! STORE !
019C 0A08              CPB      RL0,#CR        ! CK FOR END !
019E 0D0D
01A0 EE02              JR       NZ,SKPSOM      ! GOT CHAR.. !
01A2 D07C              CALR     SKPB
01A4 9E08              RET

                    !CONTROL CHAR FILTERED AND DEC LINE COUNTER !
                    SKPSOM:
01A6 0A08              CPB      RL0,#' '
01A8 2020
01AA E7F6              JR       C,LOPSTR
01AC A940              INC      R4,#1          ! GOOD CHAR !
01AE FB0C              DBJNZ    RL3,LOPSTR     ! DEC COUNT !

                    !TRUNCATE, TOO MANY CHAR !
                    LOPOVR:
01B0 D069              CALR     RECMSG
01B2 0A08              CPB      RL0,#CR        ! LOOK FOR CR !
01B4 0D0D
01B6 EEFC              JP       NZ,LOPOVR
01B8 76ED              LDA      R13,INTBUF(R14)
01BA 0000
01BC 010D              ADD      R13,#80
01BE 0050
01C0 2ED8              LDB      @R13,RL0
```

```
01C2 9E08      RET
01C4           END LINRCT

01C4           UNPACK   PROCEDURE
               !*****************************************
                *                                       *
                *  UNPACK: UNPACKS RECEIVED PACKETS     *
                *          FROM MCZ IN INTBUF AND       *
                *          LOADS IN SPECIFIED MEMORY    *
                *          AREA. ASCII CHAR ARE CON-    *
                *          VERTED TO HEX VALUES.        *
                *                                       *
                *  REG USE: INPUT  RH3 = #BYTE DATA     *
                *****************************************!
               ENTRY
01C4 A03C      LDB      RL4,RH3        ! SAVE COUNT !
01C6 DFDF      CALR     CONVAD         ! CONV START ADR !

               ! CHECK FOR USER ENTERED ADDR FOR LOAD !
01C8 0B0B      CP       R11,#%FFFE
01CA FFFE
01CC E601      JR       Z,USE_MCZADR
01CE A1B1      LD       R1,R11         ! USER SPECIFIED !

               USE_MCZADR:
01D0 76E2      LDA      R2,INTBUF(R14)
01D2 0000
01D4 A927      INC      R2,#8

               CANDS:
01D6 DFF8      CALR     TRNHEX      ! CONVERT 2-ASCII CHR !
01D8 2E18      LDB      @R1,RL0     ! STORE IN MEM !
01DA A910      INC      R1,#1
01DC FC04      DBJNZ    RL4,CANDS   ! CONV AND STORE ALL !

               ! UPDATE USER SPECIFIED ADDRESS !
01DE 0B0B      CP       R11,#%FFFE
01E0 FFFE
01E2 E601      JR       Z,NO_UPDATE    ! USE MCZ ADR !
01E4 A11B      LD       R11,R1      ! UPDATE USER ADR !

               NO_UPDATE:
01E6 9E08      RET

01E8           END UNPACK
```

```
01E8          TRNHEX PROCEDURE
              !*****************************************
               *                                       *
               *  TRNHEX: CONVERTS TWO ASCII CHAR FRM  *
               *          INTBUF TO TWO 4-BIT HEX #    *
               *          AND ADD TO CKSUM.            *
               *                                       *
               *  REG USE: INPUT  R2 = PTR TO 1ST CHR  *
               *                  RL3= CKSUM ACCUM     *
               *           RETURN R2 = UPDATE PTR      *
               *                  RL3= UPDATED ACCUM   *
               *                  RL0= HEX VALUE       *
               *                  AND C IF NON-ASCII   *
               *                     NC IF ALL GOOD    *
               *                                       *
               *****************************************!
              ENTRY
01E8 DFF6        CALR     ATOHEX        ! CONVERT 1ST CHR !
01EA 9E07        RET      C
01EC 8C8B        ADDB     RL3,RL0       ! ADD TO CKSUM !
01EE E309        SLA      R0,#12        ! MOVE TO H NIBBLE !
01F0 000C
01F2 DFFB        CALR     ATOHEX        ! CONVERT 2ND CHR !
01F4 9E07        RET      C
01F6 808B        ADDB     RL3,RL0
01F8 8408        ORB      RL0,RH0       ! COMBINE NIBBLES !
01FA 8D83        RESFLG   C
01FC 9E08        RET
01FE          END TRNHEX

01FE          ATOHEX PROCEDURE
              !*****************************************
               *                                       *
               *  ATOHEX: CONVERTS ONE ASCII CHAR TO   *
               *          4-BIT HEX NIBBLE.            *
               *                                       *
               *  REG USE: INPUT  R2 = PTR TO CHR      *
               *           RETURN R2 = PTR + 1         *
               *                  RL0= HEX NIBBLE      *
               *                                       *
               *****************************************!
              ENTRY
01FE 2028        LDB      RL0,@R2
0200 A920        INC      R2,#1          ! INC PTR !
0202 34CA        LDA      R10,R12(#CONVERT)
0204 0000*
0206 1FA0        CALL     @R10
0208 9E08        RET
020A          END ATOHEX
```

```
020A          CONVAD PROCEDURE
              !*****************************************
               *                                       *
               *  CONVAD: CONVERTS STARTING ADDRESS    *
               *          OF PACKET DATA TO HEX #.     *
               *                                       *
               *  REG USE: RETURN   R1 = ADDRESS(HEX)  *
               *                                       *
               *****************************************!
              ENTRY
020A 76E2        LDA       R2,INTBUF(R14)
020C 0000
020E D014        CALR      TRNHEX
0210 A081        LDB       RH1,RL0       ! STORE 1ST BYTE !
0212 D016        CALR      TRNHEX
0214 A089        LDB       RL1,RL0       ! STORE 2ND BYTE !
0216 9E08        RET
0218          END CONVAD

0218          CHKPAK   PROCEDURE
              !*****************************************
               *                                       *
               *  CHKPAK: CK RECEIVED MCZ PAC CKSUM    *
               *          AGAINST ACCUMULATED HEX      *
               *          VALUE CKSUM AFTER ASCII-TO-  *
               *          HEX CONVERSION.              *
               *                                       *
               *  REG USE:  RETURN  RH3 = BYTE COUNT   *
               *                    AND C IF BAD OR    *
               *                    NON-ASCII.         *
               *                                       *
               *****************************************!
              ENTRY
0218 76E2        LDA       R2,INTBUF(R14)
021A 0000
021C C303        LDB       RH3,#3
021E DFF9        CALR      CHKSUM          ! CK 1ST CKSUM !
0220 9E07        RET       C               ! BAD CK !
0222 8C34        TESTB     RH3
0224 9E06        RET       Z               ! NO DATA !
0226 93F3        PUSH      @R15,R3         ! SAVE BYTE COUNT !
0228 DFFE        CALR      CHKSUM          ! CK 2ND CKSUM !
022A 97F3        POP       R3,@R15
022C 9E08        RET
022E          END CHKPAK
```

```
022E            CHKSUM   PROCEDURE
                !*****************************************!
                *                                         *
                *  CHKSUM: CONVERTS ALL REC ASCII CHR     *
                *          IN PAC TO HEX AND ACCUM NEW    *
                *          CKSUM. COMPARE CKSUMS AND      *
                *          REPORT DIFFERENCES.            *
                *                                         *
                *  REG USE: INPUT  R2 = PTR TO PAC        *
                *                  RH3= # CHR PAIRS       *
                *           RETURN RH3= BYTE COUNT        *
                *                  RL3= NEW CKSUM         *
                *                  RH3= REC CKSUM         *
                *                  AND C IF BAD OR        *
                *                  NON-ASCII REC          *
                *                                         *
                *****************************************!
                ENTRY
022E 8CB8          CLRB     RL3            ! RESET CKSUM !
0230 D025       AB:CALR     TRNHEX         ! CONVERT PAIRS !
0232 9E07          RET      C
0234 F303          DBJNZ    RH3,AB         ! CONTINUE..... !
0236 A083          LDB      RH3,RL0
0238 93F3          PUSH     @R15,R3        ! SAVE BYTE CNT !
023A D02A          CALR     TRNHEX         ! CONVERT NEXT TWO !
023C 97F3          POP      R3,@R15
023E 9E07          RET      C
0240 8AB8          CPB      RL0,RL3        ! COMPARE CKSUMS !
0242 9E06          RET      Z              ! GOOD CK... !
0244 8D81          SETFLG   C              ! BAD CKSUM !
0246 9E08          RET
0248            END CHKSUM
```

```
                GLOBAL
0248            LOADFL PROCEDURE
                !*****************************************
                 *                                       *
                 *  LOADFL: RECEIVES PACKET FROM MCZ IN  *
                 *          FOLLOWING FORMAT:            *
                 *                                       *
                 *  <ADR><CNT><CKS1><DTA>...<DTA><CKS2>  *
                 *                                       *
                 *  ADR = START ASR IN Z8000 MEM         *
                 *  CNT = # DATA WORDS                   *
                 *  CKS1= CKSUM OF <ADR> + <CNT>         *
                 *  <DTA>...<DTA> = 30 DATA WORDS        *
                 *  CKS2= CKSUM OF DATA HEX VALUES       *
                 *                                       *
                 *  PROCEDURE VERIFIES CKSUMS BEFORE     *
                 *  STORING DATA IN Z8000 MEM. PACKETS   *
                 *  ARE ACK FOR WITH: '0' = GOOD         *
                 *                    '7' = RESEND       *
                 *                    '9' = ABORT        *
                 *  IF REC '//' FROM MCZ, ECHOS WHAT     *
                 *  REC NEXT TO CONSOLE AND ABORT.       *
                 *                                       *
                 *****************************************!

                ENTRY
0248 D125          CALR     FNAME          ! CK FILENAME !
024A 65E4          SET      MFLAGS(R14),#LDMDE !SIGNAL LOAD IN !
024C 031C
                                             !PROGRESS!
024E D109          CALR     CMDPAS         ! SND CMD TO MCZ !
0250 9E06          RET      Z              ! Z80 PROG NO LOAD !
                RECLOP:
0252 D065          CALR     LINRCT         ! GET PACKET !
0254 76E2          LDA      R2,INTBUF(R14)
0256 0000
0258 2028          LDB      RL0,@R2
025A 0A08          CPB      RL0,#'/'       ! CK FOR '//' !
025C 2F2F
025E FE10          JR       NZ,CONTIN      !NO, CONTINUE...!
0260 76E1          LDA      R1,OUTBUF(R14)   !YES,!
0262 0080
0264 2103          LD       R3,#%20
0266 0020
0268 BB21          LDIR     @R1,@R2,R3     !ERROR MSG SETUP !
026A 0310
026C 76E1          LDA      R1,OUTBUF(R14)
026F 0080
0270 0101          ADD      R1,#%20
0272 0020
```

```
0274 6FF1       LD      OUTPTR(R14),R1 !SET OUTPTR !
0276 0306

0278 34CA       LDA     R10,R12(#PBUFNC)
027A 0000*
027C 1FA0       CALL    @R10
027E 9E09       RET

                CONTIN:
0280 67E5       BIT     MFLAGS(R14),#ESC   ! CK FOR ABORT !
0282 031C
0284 EE34       JR      NZ,ABT             ! YES, ABORT...!
0286 D038       CALR    CHKPAK             ! CK CKSUMS !
0288 EF02       JR      NC,GDLD            ! GOOD LOAD !
028A D0A8       CALR    BADPAK             ! SEND NON-ACK !
028C E8E2       JR      RECLOP             ! TRY AGAIN !

                ! CHECK FOR LAST PACKET AND PRINT <ENT ADR> !
                GDLD:
028E 8CE8       CLRB    RL3
0290 8138       ADD     R8,R3    ! ACCUM NUMBER BYTES !
                                 ! OF TRANSFER !

0292 8C34       TESTB   RH3                ! CK COUNT=0 !
0294 EE28       JR      NZ,STOR            ! OK, BEGIN STR !
0296 D0AA       CALR    GODPAK             ! SEND GOOD ACK !
0298 54E0       LDL     RR0,INTBUF(R14)
029A 0000
029C 76ED       LDA     R13,OUTBUF(R14)
029E 0080
02A0 010D       ADD     R13,#%0C
02A2 000C

                ! CHECK FOR USER SPECIFIED ADDR !
02A4 0B09       CP      R9,#%AAAA
02A6 AAAA
02A8 E61D       JR      Z,END_LOAD   ! NO ECHO TO CONS !
02AA 0B0B       CP      R11,#%FFFE   ! CK FOR LOAD ADR !
02AC FFFE
02AE F608       JR      Z,SAME_ADR     ! USE MOZ ADR !
02B0 6FED       LD      OUTPTR(R14),R13  ! SET OUTBUF ADR !
02B2 0306
02B4 61E5       LD      R5,ADR_STR(R14) ! GET USER ADR !
02B6 0344
02B8 76CA       LDA     R10,CONVW(R12)
02BA 0000*
02BC 1FA0       CALL    @R10       ! CONVERT TO ASCII AND !
                                   ! AND STORE IN OUTBUF !
02BE F801       JR      FIN_BUF
                SAME_ADR:
```

```
02C0 1DD0       LDL     @R13,RR0

                FIN_BUF:
02C2 3402       LDAR    R2,ENTADR    !LOAD ENTRY LABEL!
02C4 0040
02C6 76E1       LDA     R1,OUTBUF(R14)
02C8 0080
02CA 2100       LD      R0,#6
02CC 0006
02CE BB21       LDIR    @R1,@R2,R0
02D0 0010
02D2 76ED       LDA     R13,OUTBUF(R14)
02D4 0080
02D6 010D       ADD     R13,#%10
02D8 0010
02DA 6FED       LD      OUTPTR(R14),R13
02DC 0306
02DE 34CA       LDA     R10,R12(#PRNTEF)
02E0 0000*
02E2 1FA0       CALL    @R10          ! PRINT MESSAGE !
                END_LOAD:
02E4 9E08       RET

                STOR:
02E6 D06F       CALR    CONVAD
02E8 D0D3       CALR    GODPAK        ! SEND ACK !
02EA D094       CALR    UNPACK        ! UNPACK AND STORE !
02EC E8B2       JR      RECLOP        ! CONTINUE....!

                ABT:
02EE 3402       LDAR    R2,EMSG
02F0 000A
02F2 34CA       LDA     R10,R12(#SNDMSG)
02F4 0000*
02F6 1FA0       CALL    @R10           ! SEND MESSAGE !
02F8 D0DD       CALR    ABORTM         ! SEND ABORT !
02FA 9E08       RET

02FC            END LOADFL

                EMSG:
02FC 07            BVAL   7
02FE 2F41          WVAL   '/A'
0300 424F          WVAL   'BO'
0302 5254          WVAL   'RT'
0304 0D            BVAL   %0D
                ENTADR:
0306 454E          WVAL   'EN'
0308 5452          WVAL   'TR'
030A 5920          WVAL   'Y '
```

```
030C 504F      WVAL  'PO'
030E 494E      WVAL  'IN'
0310 5420      WVAL  'T '

               END LOAD_CMD
```

4. REGISTER MODULE

```
Z8000ASM  2.02
LOC     OBJ CODE     STMT SOURCE STATEMENT

                        1 REGISTER_CMD MODULE
                    $LISTON  $TTY
                    !*****************************************
                     *                                       *
                     *  REGISTER CMD: DISPLAYS CONTENTS OF   *
                     *                ALL USER (') REGS OR   *
                     *    EACH REGISTER INDIVIDUALLY IN THE  *
                     *    SUBSTITUTION MODE. A 'Q' ENDS THE  *
                     *    SESSION; A CR ADVANCES IN ORDER    *
                     *    THROUGH REG CONTENTS WITHOUT ANY   *
                     *    CHANGES; AND A VALID HEX ENTRY     *
                     *    WILL ALTER THE REGISTER CONTENTS.  *
                     *                                       *
                     *  SYNTAX:  REGISTER [<REG NAME>]       *
                     *                                       *
                     *****************************************!


                    EXTERNAL  EROR    LABEL
                    EXTERNAL  GETCHR  PROCEDURE
                    EXTERNAL  STOBUF  PROCEDURE
                    EXTERNAL  DISP    PROCEDURE
                    EXTERNAL  PRNTBF  PROCEDURE
                    EXTERNAL  CONVB   LABEL
                    EXTERNAL  GETNXT  PROCEDURE
                    EXTERNAL  DISPNC  LABEL
                    EXTERNAL  CONVW   PROCEDURE


                    $SECTION REGISTER_PROC
                    $REL 0

0000 30  20           LKTBL ARRAY [* BYTE]:='0 1 2 3 4 5
                          6 7 8 9 10111213141 5PCFCRLRH'

0002 31  20
0004 32  20
0006 33  20
0008 34  20
000A 35  20
000C 36  20
000E 37  20
```

```
0010 38  20
0012 39  20
0014 31  30
0016 31  31
0018 31  32
001A 31  33
001C 31  34
001E 31  35
0020 50  43
0022 46  43
0024 52  43
0026 52  48

            GLOBAL
              PRREG2   LABEL
0028          PRREG1   PROCEDURE
            !*****************************************
             *                                       *
             *  PRREG1: OUTPUT CONTENTS OF USER      *
             *          REGISTERS 0-13 TO CONS.      *
             *                                       *
             *  PRREG2: OUTPUTS CONTENTS OF USER     *
             *          REGS R14,R15,RPC,RFC.        *
             *                                       *
             *****************************************!
            ENTRY
0028 BD6E     LDK       R6,#14          ! SET # REGS !
002A 34F1     LDA        R1,R14(#R0_)       ! LOAD BASE ADR !
002C 031E
002E F803     JR        PLOOP

            ! PRREG2 ENTRY POINT !
            PRREG2:
0030 BD64     LDK       R6,#4    ! SET 2ND LINE REG # !
0032 34F1     LDA        R1,R14(#R14_)  ! LOAD BASE ADR !
0034 033A


            ! MAIN PRINT LOOP !
            PLOOP:
0036 2115     LD        R5,@R1
0038 34C4     LDA       R10,R12(#CONVW)   ! CONVERT HEX TO !
003A 0000*
003C 1FA0     CALL      @R10
                                        ! ASCII !
003E AB60     DEC       R6,#1           ! DONE? !
0040 E604     JR        Z,PRNT          ! OUTPUT IF YES !
0042 69E0     INC       OUTPTR(R14),#1
0044 0306
0046 A911     INC       R1,#2           ! GET NEXT REG ADR !
0048 F8F6     JR        PLOOP           ! LOOP !
```

```
                PRNT:
004A 34CA         LDA       R10,R12(#PRNTBF)
004C 0000*
004E 1EA8         JP        @R10

0050              END PRREG1

                  GLOBAL
                     RGHDR2    LABEL
0050                 RGHDR1    PROCEDURE
                  !*****************************************
                   *                                       *
                   *  RGHDR1: OUTPUT HEADER FOR REGISTERS  *
                   *          R0 - R13.                    *
                   *                                       *
                   *  RGHDR2: OUTPUT HEADER FOR REGISTERS  *
                   *          R14, R15, RPC, RFC.          *
                   *                                       *
                   *****************************************!
                  ENTRY
0050 BD0E         LDK       R0,#14      ! SET PRINT COUNT !
0052 3405         LDAR      R5,LKTBL
0054 FFAA
0056 E805         JR        HLOOP

                  ! SECOND HEADER ENTRY POINT !
                  RGHDR2:
0058 BD04         LDK       R0,#4       ! SET PRINT COUNT !
005A 3405         LDAR      R5,LKTBL
005C FFA2
005E 0105         ADD       R5,#28
0060 001C

                  ! MAIN PRINT LOOP !
                  HLOOP:
0062 31E2         LD        R2,R14(#OUTPTR)   ! LOAD OUTBUF INDEX !
0064 0306
0066 0C25         LDB       @R2,#'R'
0068 5252
006A 69E0         INC       OUTPTR(R14)
006C 0306
006E 2151         LD        R1,@R5      ! GET CHR FROM TABLE !
0070 DFFA         CALR      STOBUF      ! STORE IN OUTBUF !
0072 AB00         DEC       R0,#1       ! DONE? !
0074 E6EA         JR        Z,PRNT      ! YES, OUTPUT BUFFER !
0076 69E1         INC       OUTPTR(R14),#2  ! NO, CONTINUE!
0078 0306
007A A951         INC       R5,#2       ! ADVANCE TABLE INDEX!
007C E8F2         JR        HLOOP
```

```
007E            END  RGHDR1

                GLOBAL
007E            STOBUF  PROCEDURE
                !*****************************************
                 *                                       *
                 *  STOBUF: STORES CONTENTS OF REGISTER  *
                 *          INTO OUTBUF AND INCREMENTS   *
                 *          OUTPTR.                      *
                 *                                       *
                 * REG USE: INPUT  R1 = CONTENTS         *
                 *                                       *
                 *****************************************!
                ENTRY
007E 93F4          PUSH     @R15,R4       ! SAVE WORK REG !
0080 61E4          LD       R4,OUTPTR(R14)
0082 0306
0084 2E41          LDB      @R4,RH1       ! STR 1ST BYTE !
0086 A940          INC      R4
0088 2E49          LDB      @R4,RL1       ! STR 2ND BYTE !
008A 69E1          INC      OUTPTR(R14),#2
008C 0306
008E 97F4          POP      R4,@R15       ! RESTORE WRK REG !
0090 9E08          RET
0092            END STOBUF

                GLOBAL
0092            REGISTER  PROCEDURE
                !*****************************************
                 *                                       *
                 *  REGISTER: DISPLAY AND ALLOW CHANGE   *
                 *            OF USER REGISTERS.         *
                 *                                       *
                 *  REG USE: INPUT  ALL                  *
                 *           RETURN SAME                 *
                 *                                       *
                 *****************************************!
                ENTRY
0092 340B          LDAR     R11,LKTBL
0094 FF6A
0096 34CA          LDA      R10,R12(#GETNXT)
0098 0000*
009A 1FA0          CALL     @R10          ! SKIP REST CMD !
009C EF01          JR       NZ,AM
009E E85B          JR       PRNTAL

00A0 0A08       AM:CPB       RL0,#'R'
00A2 5252
00A4 EE22          JR       NZ,GLOBER
                ! CHECK DISPLAY MODE - R, RR, RH, RL !
```

```
00A6 34CA      LDA       R10,R12(#GETCHR)
00A8 0000*
00AA 1FA0      CALL      @R10           ! GET NXT CHR !
00AC E61E      JR        Z,GLOBER       ! ERROR !
00AE A08E      LDB       RL6,RL0        ! GET FIRST CHR !
00B0 0A09      CPB       RL0,#'R'       ! CK FOR 'RR' !
00B2 5252
00B4 E608      JR        Z,REGID        ! YES, GET REG ID !
00B6 8C68      CLRB      RH6
00B8 0A28      CPB       RL0,#'L'       ! IS LOW BYTE? !
00BA 4C4C
00BC E604      JR        Z,REGID        ! YES, GET REG ID !
00BE C602      LDB       RH6,#2
00C0 0A08      CPB       RL0,#'H'       ! IS HIGH BYTE? !
00C2 4848
00C4 EE04      JR        NZ,SAVID

               ! GET REGISTER ID NUMBER !
               REGID:
00C6 34CA      LDA       R10,R12(#GETCHR)
00C8 0000*
00CA 1FA0      CALL      @R10           ! GET 1ST CHAR !
00CC E60E      JR        Z,GLOBER
               SAVID:
00CE A081      LDB       RH1,RL0
00D0 34CA      LDA       R10,R12(#GETCHR)
00D2 0000*
00D4 1FA0      CALL      @R10           ! GET 2ND CHR !
00D6 EE02      JR        NZ,SAVID2
00D8 C920      LDB       RL1,#' '       ! PAD WITH SPACE !
00DA E801      JR        FNDNAM
               SAVID2:
00DC A089      LDB       RL1,RL0        ! SAVE BOTH DIGITS !
               ! FIND REG ID IN REG LOOKUP TABLE !
               FNDNAM:
00DE 2108      LD        R8,#18         ! TABLE LENGTH !
00E0 0012
00E2 41B9      LD        R9,R11    ! BASE OF LKUP TBL !
00E4 BB94      CPIR      R1,@R9,R8,EQ ! LOOK FOR MATCH !
00E6 0816

00E8 E603      JR        Z,AJ

               GLOBER:
00EA 34CA      LDA       R10,R12(#EROR)
00EC 0000*
00EE 1EA8      JP        @R10

00F0 AB91  AJ:DEC        R9,#2          ! ADJUST CORRECT ADR!
00F2 2102      LD        R2,#17         ! DETERMINE REG MEM !
```

```
00F4 0011
00F6 8382       SUB     R2,R8           ! INDEX !
00F8 A128       LD      R8,R2
00FA B381       SLL     R8,#1
00FC 0001
                ! DETERMINE DISPLAY MODE OUTLINE !
00FE 0A0E       CPB     RL6,#'H'        ! RL6=MODE !
0100 4848
0102 E65E       JR      Z,BYT           ! BYT MODE !
0104 0A0E       CPB     RL6,#'L'
0106 4C4C
0108 E65B       JR      Z,BYT           ! ALSO BYTE MODE !
010A 0A0E       CPB     RL6,#'R'
010C 5252
010E E628       JR      Z,LWORD         ! LONG WORD MODE !

                ! SINGLE WORD DISPLAY MODE ROUTINE !
                SWORD:
0110 2101       LD      R1,#'R '
0112 5220
0114 D04C       CALR    STOBUF          ! STORE IN OUTBUF !
0116 6BE0       DEC     OUTPTR(R14),#1
0118 0306
011A 2191       LD      R1,@R9          ! STORE REG ID !
011C D050       CALR    STOBUF
011E 69E0       INC     OUTPTR(R14),#1      ! ADD SPACE !
0120 0306
0122 34ED       LDA      R13,R14(#R0_)
0124 031E
0126 71D5       LD      R5,R13(R8)
0128 0800
012A 34CA       LDA     R10,R12(#DISP)
012C 0000*
012E 1FA0       CALL    @R10            ! OUTPUT TO CONS !
0130 9E07       RET     C               ! REC 'Q' !
0132 E60A       JR      Z,SAME          ! REC CR, STAYS SAME !
0134 2192       LD      R2,@R9
0136 34BA       LDA     R10,R11(#34)
0138 0022
013A 8BA2       CP      R2,R10
013C EE01       JR      NZ,JMPOVR       ! NOT PC REG !
013E A330       RES     R3,#0           ! CHG TO EVEN ADR !

                JMPOVR:
0140 76ED       LDA     R13,R0_(R14)    ! CHG REG CONTENTS !
0142 031E
0144 73D3       LD      R13(R8),R3
0146 0800
                SAME:
0148 A981       INC     R8,#2           ! ADJ REG INDEX !
```

```
314A A991      INC      R9,#2          ! ADJ TABLE ADR !
014C 34B3      LDA      R3,R11(#35)    ! GET END OF TBLE !
014E 0023
0150 8B39      CP       R9,R3          ! END OF TABLE? !
0152 F7DE      JR       C,SWORD        ! NO, CONTINUE..!
0154 9E08      RET                     ! YES, DONE !

               ! ROUTINE TO PRINT ALL HEADERS AND CONTENTS !
               PRNTAL:
0156 D084      CALR     RGHDR1
0158 D099      CALR     PRREG1
015A D082      CALR     RGHDR2
015C D097      CALR     PRREG2
015E 9E08      RET

               ! LONG WORD DISPLAY MODE ROUTINE !
               LWORD:
0160 A1BD      LD       R13,R11  ! R15 ADDR !
0162 010D      ADD      R13,#30
0164 001E
0166 8BD9      CP       R9,R13
0168 EFC0      JR       NC,GLOBER      ! RR0 - RR14 OK !
016A 0A09      CPB      RL1,#' '       ! CK IF DIGIT REG !
016C 2020
016E E602      JR       Z,EVNCK
0170 A710      BIT      R1,#0     ! CK IF 2ND CHR EVEN !
0172 E801      JR       OVER
               EVNCK:
0174 A718      BIT      R1,#8
               OVER:
0176 EEB9      JR       NZ,GLOBER

               ! LONG WORD DISPLAY MODE ROUTINE !
               LWLOOP:
0178 2101      LD       R1,#'RR'
017A 5252
017C D080      CALR     STOBUF       ! STORE IN OUTBUF !
017E 2191      LD       R1,@R9
0180 1FA0      CALL     @R10     ! ADD REG ID !
0182 69E0      INC      OUTPTR(R14),#1     ! ADD SPACE !
0184 0306
0186 34E4      LDA      R4,R14(#R0_)    ! LOAD 1ST WORD !
0188 031E
018A 7145      LD       R5,R4(R8)
018C 0800
018E 34CA      LDA      R10,R12(#CONVW)
0190 0000*
0192 1FA0      CALL     @R10          ! CONVERT 1ST !
0194 A941      INC      R4,#2
0196 7145      LD       R5,R4(R8) ! LOAD 2ND WORD !
```

```
0198 0800
019A 34CA       LDA     R13,R12(#DISP)
019C 0000*
019F 1FA0       CALL    @R10
01A0 9E07       RET     C
01A2 E607       JR      Z,AROUND    ! REC CR, NO CHG !
01A4 34E4       LDA      R4,R14(#R0_)
01A6 031E
01A8 7342       LD      R4(R8),R2
01AA 0800
01AC A941       INC     R4,#2
01AE 7343       LD      R4(R8),R3
01B0 0800

            AROUND:
01B2 A983       INC     R8,#4
01B4 A993       INC     R9,#4       ! INCREMENT TO NEXT RR !
01B6 34B3       LDA      R3,R11(#16)
01B8 0010
01BA 8B39       CP      R9,R3       ! FINISHED? !
01BC E7DD       JR      C,LWLOOP
01BE 9E08       RET                 ! DONE !

            ! BYTE DISPLAY ROUTINE !
            BYT:
01C0 A1B4       LD      R4,R11
01C2 A947       INC      R4,#8
01C4 8B49       CP       R9,R4          ! CK IF REG >R7 !
01C6 EF91       JR       NC,GLOBER
01C8 B361       SRL      R6,#8
01CA FFF8
01CC EE01       JR       NZ,THRU
01CE A980       INC      R8

            THRU:
01D0 A1B4       LD      R4,R11
01D2 0104       ADD      R4,#18
01D4 0012
01D6 7141       LD       R1,R4(R6)
01D8 0600
01DA D0AF       CALR     STOBUF       ! STORE IN OUTBUF !
01DC 2191       LD       R1,@R9
01DE 1FA0       CALL     @R10
01E0 31E4       LD       R4,R14(#R0_)
01E2 031E
01E4 704D       LDB      RL5,R4(R8)
01E6 0800
01E8 34CA       LDA      R10,R12(#CONVB)
01EA 0000*
01EC 1FA0       CALL     @R10          ! CONVERT WORD !
```

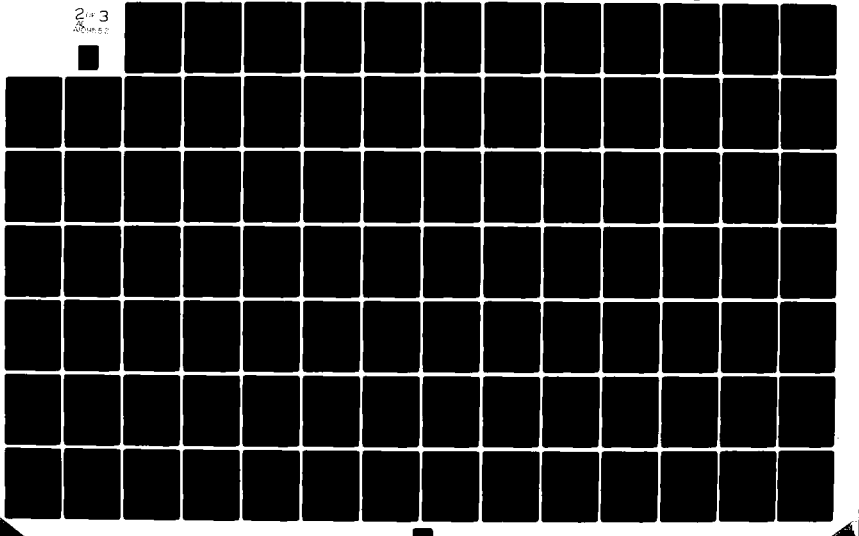
```
01EE 69E0      INC       OUTPTR(R14),#1
01F0 03C6
01F2 34CA      LDA       R10,R12(#DISPNC)
01F4 0000*
01F6 1FA0      CALL      @R10     ! OUTPUT, CK FOR INPUT !
01F8 9E07      RET       C          ! FINISHED, REC 'Q' !
01FA E604      JR        Z,ABA     ! REC CR, STAYS SAME !
01FC 34E4      LDA        R4,R14(#R0_)
01FE 031E
0200 724B      LDB          R4(R8),RL3
0202 0800

               ABA:
0204 A980         INC       R8,#1
0206 AB61         DEC       R6,#2        ! INCREMENT INDEXES !
0208 E6E3         JR        Z,THRU       ! OUTPUT LOW BYTE !
020A 2106         LD        R6,#2
020C 0002
020E A991         INC        R9,#2      ! GET NEXT BYTE GRP !
0210 A1B3         LD       R3,R11
0212 A937         INC        R3,#8
0214 8B39         CP        R9,R3        ! CK FOR END !
0216 E7DC         JR        C,THRU       ! NO, CONTINUE...!
0218 9E08         RET                    ! YES. !

021A           END REGISTER
               END REGISTER_CMD
```

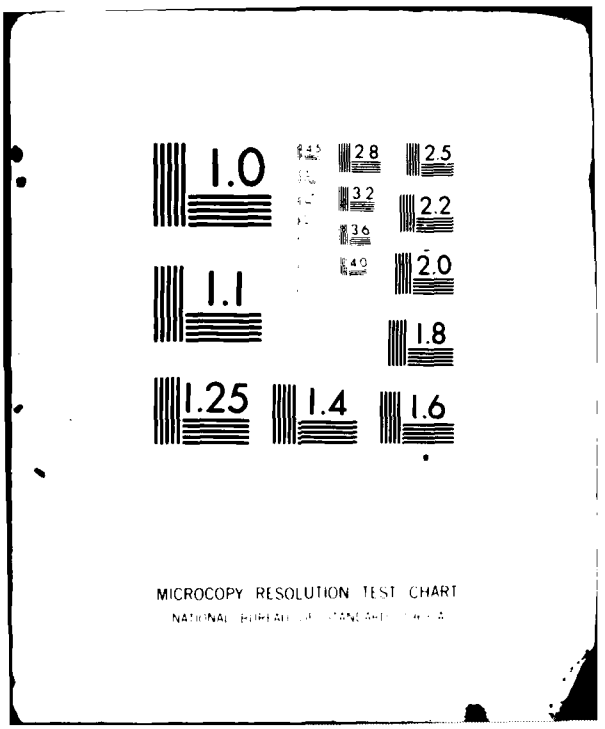
5. DISPLAY MODULE

```
Z8000ASM  2.02
LOC    OBJ CODE     STMT SOURCE STATEMENT

                      1 DISPLAY_CMD MODULE
                $LISTON  $TTY

                !*****************************************
                 *                                       *
                 *   DISPLAY CMD: DISPLAYS CONTENTS OF   *
                 *                SPECIFIED MEMORY LOC   *
                 *      WITH ABILITY TO CHANGE CONTENTS  *
                 *      OF MEM.AS PER <CNT>. CONTENTS    *
                 *      ARE DISPLAYED 16-BYTES/LINE WITH*
                 *      SPACE BETWEEN BYTES/WORDS/LWORDS*
                 *      AS SELECTED [B/W/L]. WITHOUT     *
                 *      COUNT,CONTENT IS DISPLAYED IN    *
                 *      UNIT B/W/L. ENTERED DATA WILL    *
                 *      CHG CONTENTS; CR WILL NOT; AND   *
                 *      'Q' WILL EXIT CMD.               *
                 *                                       *
                 *      FILL: STORES GIVEN DATA(WORD)    *
                 *         IN ALL INCLUSIVE MEMORY LOC   *
                 *         DEFINED BY <BGN ADR> AND      *
                 *         <END ADR>.                    *
                 *                                       *
                 *      MOVE: MOVES BLOCK OF DATA AS     *
                 *         DEFINED BY SIZE <SIZ> FROM    *
                 *         START ADR <SCR> TO DEST.      *
                 *         ADR <DST>.                    *
                 *                                       *
                 *  SYNTAX: DISPLAY <ADR>[<CNT>][B/W/L]  *
                 *          MOVE <SCR> <DST> <SIZ>       *
                 *          FILL <BGN ADR><END ADR><WD>  *
                 *                                       *
                 *****************************************!

                CONSTANT

                ( INCLUDE GLOBAL CONSTANTS )

                EXTERNAL  EROR      LABEL
                EXTERNAL  ASCHEX    PROCEDURE
```

```
                GLOBAL
                $SECTION DISPLAY_PROC
                $REL 0


                GLOBAL
0000            SKPBLK  PROCEDURE
                !*****************************************
                 *                                       *
                 *  SKPBLK: SKIP OVER BLANKS TO NEXT     *
                 *          CHARACTER.                   *
                 *                                       *
                 *  REG USE:  RETURN  RL0 = 1ST NON-BLK  *
                 *                    CHAR AND Z IF =CR  *
                 *                                       *
                 *****************************************!
                ENTRY
                ! SKIP OVER BLANKS TO NEXT ARGUMENT !
0000 DF30          CALR     GETCHR
0002 9F06          RET      Z              ! GOT CR !
0004 0A08          CPB      RL0,#' '       ! CK FOR BLANK !
0006 2020
0008 E6FB          JR       Z,SKPBLK       ! YES,..... !
000A 9F08          RET                     ! GOT CHAR !

000C            END SKPBLK

                GLOBAL
000C            GETADR  PROCEDURE
                !*****************************************
                 *                                       *
                 *  GETADR: GETS NEXT ARGUMENT AND       *
                 *          CONVERTS TO HEX ADDRESS.     *
                 *                                       *
                 *  REG USE: INPUT  RL0 = 1ST CH OF ARG  *
                 *           RETURN  R3 = HEX ADR        *
                 *                   AND Z,C  IF CR ONLY*
                 *                      Z,NC IF ARG,CR   *
                 *                     NZ,NC IF ARG,SP   *
                 *                                       *
                 *****************************************!
                ENTRY
                ! CK FOR CR ONLY !
000C 8D38          CLR      R3
000E 0A08          CPB      RL0,#CR        ! CK FOR CR !
0010 0D0D
0012 EE02          JR       NZ,NOTCR
0014 8D81          SETFLG   C
0016 9E08          RET                     ! RETURN FOR CR ONLY !
```

AD-A109 552 NAVAL POSTGRADUATE SCHOOL MONTEREY CA F/G 9/2
SASS HARDWARE ARCHITECTURE AND DEVELOPMENTAL MONITOR.(U)
JUN 81 G S BAKER
UNCLASSIFIED NPS52-81-016 NL




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
MICROCOPY RESOLUTION TEST CHART

```
                    !CONVERT ASCII ADDRESS TO HEX ADDRESS !
                    NOTCR:
0018 DFE2              CALR     CONVERT      !BYTE TO 4-BIT HEX !
001A E70A              JR       C,REPERR     ! GOT BAD CHR !
001C BFB8              RLDB     RL0,RL3
001F BE38              RLDB     RL0,RH3      !SHFT LEFT TO MSW !
0020 DFF0              CALR     GETCHR       ! GET CHR FROM INTBUF!
0022 9E06              RET      Z
0024 0A08              CPB      RL0,#' '     !CK FOR SPACE !
0026 2020
0028 EEF7              JR       NZ,NOTCR     ! IF NOT, CONT.....!
002A D016              CALR     SKPBLK       ! SKIP TO NEXT ARG !
002C 8D83              RESFLG   C
002E 9E08              RET                   ! SPACE AFTER ARG !

                    REPERR:
0030 34CA              LDA      R10,R12(#ERCR)
0032 0000*
0034 1FA8              JP       @F10
0036                END GETADR

                    GLOBAL
0036                GETNXT  PROCEDURE
                    !*****************************************
                     *                                       *
                     *  GETNXT: SKIP TO BEGINING OF NEXT     *
                     *          ARGUMENT IN COMMAND.         *
                     *                                       *
                     *  REG USE:  RETURN  RL0 = CHAR OR CR   *
                     *                    AND Z IF = CR      *
                     *                                       *
                     *****************************************!
                    ENTRY
                    ! SKIP OVER CURRENT ARGUMENT TO NEXT SPACE !
0036 DFFB              CALR     GETCHR
0038 9E06              RET      Z            ! RTN IF CH=CR !
003A 0A08              CPB      RL0,#' '     ! FIND FIRST SPACE !
003C 2020
003E EEFB              JR       NZ,GETNXT
0040 E8DF              JR       SKPBLK       ! NOW SKIP BLANKS !

0042                END GETNXT
```

```
                GLOBAL
0042            GETCHR  PROCEDURE
                !*****************************************
                 *                                       *
                 *  GETCHR: GETS NEXT CHR FROM INTBUF    *
                 *          AND INCREMENTS INTPTR.       *
                 *                                       *
                 *  REG USE: RETURN  RL0 = CHR           *
                 *                   AND Z IF CR         *
                 *                                       *
                 *****************************************!
                ENTRY
0042 93F2          PUSH     @R15,R2       ! SAVE WORK REG !
0044 61E2          LD       R2,INTPTR(R14)
0046 0304
0048 2028          LDB      RL0,@R2       ! GET CHR !
004A 69E0          INC      INTPTR(R14),#1    ! INC PTR !
004C 0304
004E 0A08          CPB      RL0,#CR        !CK FOR CR !
0050 0D0D
0052 97F2          POP      R2,@R15
0054 9E08          RET
0056            END GETCHR

                GLOBAL
0056            CONVERT  PROCEDURE
                !*****************************************
                 *                                       *
                 *  CONVERT: CONVERTS 8-BIT ASCII CHR    *
                 *           TO 4-BIT HEX VALUE. VALID   *
                 *           CHR IS 0-9 OR A-F; IF NOT   *
                 *           CHR, EXIT TO EXEC EROR.     *
                 *                                       *
                 *  REG USE: INPUT  RL0 = 8-BIT ASCII    *
                 *           RETURN RL0 = 4-BIT HEX IN   *
                 *                        LSW.           *
                 *                                       *
                 *****************************************!
                ENTRY
                ! CHECK FOR VALID CHAR !
0056 0A08          CPB      RL0,#'0'  ! FILTER <'0' ASCII !
0058 3030
005A 9E07          RET      C         ! ERROR !
005C 0A08          CPB      RL0,#'9'+1   ! CK IF DIGIT !
005E 3A3A
0060 E708          JR       C,NOFIX
0062 0A08          CPB      RL0,#'A'  ! FILTER <'A' ASCII !
0064 4141
0066 9E07          RET      C
0068 0A08          CPB      RL0,#'F'+1 !FILTER >'F' ASCII !
```

```
006A 4747
006C EF06       JR      NC,RETSIG   !ERROR!
006E 0208       SUBB    RL0,#7      ! ALPHA ADJUST !
0070 0707
                NOFIX:
0072 0608       ANDB    RL0,#%0F    ! GET LOW NIBBLE !
0074 0F0F
0076 8D83       RESFLG  C
0078 9E08       RET                 ! RTN HEX VALUE !
                RETSIG:
007A 8D81       SETFLG  C
007C 9E08       RET                 ! RTN FOR BAD CHR !

007E            END CONVERT

                GLOBAL
                   PBUFNC  LABEL
007E               PRNTBF  PROCEDURE
                !*****************************************
                 *                                       *
                 *  PRNTBF: PRINT CONTENTS OF OUTBUF     *
                 *          TO CONS WITH CR AT END.      *
                 *                                       *
                 *  PBUFNC: PRINT BUFFER CONTENTS WITH   *
                 *          NO CR.                       *
                 *                                       *
                 *****************************************!
                ENTRY
                ! STORE CR IN OUTBUF !
007E 61E2       LD      R2,OUTPTR(R14)
0080 0306
0082 0C25       LDB     @R2,#CR
0084 0D0D
0086 69E0       INC     OUTPTR(R14),#1
0088 0306

                PBUFNC:
008A 76E1       LDA      R1,OUTBUF(R14)   ! LOAD ADR OF OUTBUF !
008C 0080
                ! OUTPUT LOOP !
                PRNT:
008E 2018       LDB     RL0,@R1         ! GET CHR !
0090 A910       INC     R1              ! INC INDEX !
0092 DFE9       CALR    SNDCHR          ! OUTPUT CHR !
0094 E604       JR      Z,CUTLF         ! ?CHR = CR !
0096 4BE1       CP      R1,OUTPTR(R14)     ! CK FOR END !
0098 0306
009A E7F9       JR      C,PRNT          ! LOOP.....!
009C E802       JR      FINI            ! FINISHED !
```

```
                ! ADD LF AFTER OUTPUT OF CR !
                OUTLF:
009E C80A          LDB      RL0,#LF        ! OUTPUT LF !
00A0 DFF0          CALR     SNDCHR

                ! FILL OUTBUF WITH BLANKS AND RESET OUTPTR !
                FINI:
00A2 76E3          LDA      R3,OUTBUF(R14)
00A4 0080
00A6 6FE3          LD       OUTPTR(R14),R3 ! RESET PTR !
00A8 0306
00AA 2100          LD       R0,#OUTSIZ/2-1 ! FILL CNT !
00AC 003F
00AE 4DE5          LD       OUTBUF(R14),#'  '
00B0 0080
00B2 2020
00B4 76E2          LDA       R2,OUTBUF(R14)
00B6 0080
00B8 A123          LD       R3,R2
00BA A931          INC      R3,#2
00BC BB21          LDIR     @R3,@R2,R0     ! FILL OUTBUF !
00BE 0030
00C0 9E08          RET
00C2           END PRNTBF

               GLOBAL
00C2           SNDCHR  PROCEDURE
               !*****************************************
                *                                       *
                *  SNDCHR: CK MONITOR FLAG WORD FOR     *
                *          OUTPUT STOP SIGNAL (OSTOP);  *
                *          IF NOT, SEND CHAR TO CONS.   *
                *                                       *
                *  REG USE: INPUT   RL0= CHR            *
                *           RETURN  RL0= CHR AND Z IF   *
                *                        CHR = CR.      *
                *                                       *
                *****************************************!
               ENTRY
               ! WAIT FOR OUTPUT OK SIGNAL !
00C2 67E2          BIT      MFLAGS(R14),#OSTOP   ! CK FLAG !
00C4 031C
00C6 EEFD          JR       NZ,SNDCHR
               !OUTPUT CHAR TO TERMINAL !
00C8 3A04          INB      RH0,PORTBC    ! GET PORT STATUS !
00CA FFE3
00CC A600          BITB     RH0,#TXR       ! TRANS RDY? !
00CE E6F9          JR       Z,SNDCHR      ! NO, CONTINUE...!
00D0 3A86          OUTB     PORTBD,RL0    ! YES, OUTPUT CHR !
00D2 FFE1
```

```
00D4 0A08       CPB     RL0,#CR
00D6 0D0D
00D8 9E08       RET
00DA          END SNDCHR

              GLOBAL
                 CONVB   LABEL
00DA             CONVW   PROCEDURE
              !*****************************************
               *                                       *
               *  CONVW: CONVERT INTERNAL WORD, 4-     *
               *         4-BIT HEX VALUES TO FOUR      *
               *         8-BIT ASCII REPRESENTATIONS   *
               *         OF THE HEX VALUES.            *
               *                                       *
               *  CONVB: CONVERT INTERNAL BYTE HEX     *
               *         VALUE TO ASCII CHARACTERS.    *
               *                                       *
               *  REG USE: INPUT  R5 = WORD/BYTE(S)    *
               *                  R3 = CKSUM ACCUM     *
               *           RETURN R3 = UPDATED ACCUM   *
               *                       AND ASCII CHR   *
               *                       IN OUTBUF       *
               *                                       *
               *****************************************!
              ENTRY
              ! CONVERT WORD !
00DA A050        LDB       RH0,RH5      ! 1ST BYTE !
00DC DFFF        CALR      NIBBLE
              ! CONVERT BYTE ENTRY POINT !
              CONVB:
00DE A0D0        LDB       RH0,RL5

              NIBBLE:
00E0 BE08        RLDB      RL0,RH0      ! FIRST NIBBLE !
00E2 DFFF        CALR      CONPUT
00E4 BE08        RLDB      RL0,RH0       ! NEXT NIBBLE !
              !CONVERT NIBBLE TO ASCII CHAR AND STORE !
              CONPUT:
00E6 0608        ANDB      RL0,#%0F     ! GET NIBBLE !
00E8 0F0F
00EA 8C8B        ADDB      RL3,RL0      ! UPDATE CKSUM !
00EC 0A08        CPB       RL0,#%0A     ! 0-9? !
00EE 0A0A
00F0 E702        JR        C, ASCII     ! YES... !
00F2 0008        ADDB      RL0,#7       ! NO, CONVERT CHR !
00F4 0707
              ASCII:
00F6 0008        ADDB      RL0,#%30     ! CONVERT TO ASCII !
00F8 3030
```

```
                ! STORE IF OUTBUF !
00FA 93F1          PUSH     @R15,R1       ! SAVE R1 !
00FC 61E1          LD       R1,OUTPTR(R14)
00FE 0306
0100 2E18          LDB      @R1,RL0       ! STORE CHR !
0102 69E0          INC      OUTPTR(R14)
0104 0306
0106 97F1          POP      R1,@R15
0108 9E08          RET
010A            END CONVW

                GLOBAL
                   DISPNC   LABEL
010A               DISP     PROCEDURE
                !*****************************************
                 *                                       *
                 *  DISP: CONVERTS INPUT TO FOUR ASCII   *
                 *        CHAR, STORES IN OUTBUF, AND    *
                 *        DISPLAYS WITH CR; GETS NEXT    *
                 *        8 ASCII CHARS, STORES IN       *
                 *        INTBUF, AND CONVERTS TO HEX    *
                 *        LONG WORD IN RR2.              *
                 *                                       *
                 *  DISPNC: SENDS TO CONSOLE ALL IN      *
                 *        OUTBUF UP TO OUTPTR, WITHOUT   *
                 *        CR; GETS NEXT 8-ASCII CHR      *
                 *                                       *
                 *  REG USE: INPUT   R5 = WORD           *
                 *           RETURN  R2,R3 = 8-HEX       *
                 *                   AND C    IF REC 0   *
                 *                       NC,Z IF CR      *
                 *                      NC,NZ IF INPUT   *
                 *                       WITH CR         *
                 *                                       *
                 *****************************************!
                ENTRY
010A D019          CALR     CONVW         ! ADD SPACE !
010C 69E0          INC      OUTPTR(R14),#1
010E 0306

                DISPNC:
0110 D044          CALR     PBUFNC        ! OUTPUT TO CONS !
0112 34CA          LDA      R10,R12(#ASCHEX)
0114 0000*
0116 1FA0          CALL     @R10     ! GET NEW R2,R3 INPUT !
0118 9E08          RET

011A            END DISP
```

```
                GLOBAL
011A            DISPLAY PROCEDURE
                !*****************************************
                 *                                       *
                 *  DISPLAY: DISPLAYS SPECIFIED CONTENTS*
                 *           OF MEMORY.                  *
                 *                                       *
                 *  REG USE: RH7,RL7 = #BYTES/SPACE      *
                 *           R8 = OUTBUF ADR FOR OUTPUT  *
                 *           R11 = <ADDR>                *
                 *           R13 = <COUNT>               *
                 *                                       *
                 *****************************************!
                ENTRY
011A D073          CALR     GETNXT
011C 2107          LD       R7,#%202
011E 0202
0120 EE01          JR       NZ,GOTARG      ! HAVE NEXT ARG !
0122 F84C          JR       EXERR          ! RETURN TO EXEC !
                GOTARG:
0124 D08D          CALR     GETADR         ! NEXT ARG IS ADR !
0126 A13B          LD       R11,R3         ! SAVE ADDRESS !
0128 F637          JR       Z,SUBMOD       ! <ADR><CR> = SUBMODE!
012A D090          CALR     GETADR         ! GET <CNT> !
012C A13D          LD       R13,R3
012E F601          JR       Z,TSTCNT       ! GOT CR AFTER <CNT> !
0130 EE45          JR       NZ,EXERR       ! NOT THERE....!

                TSTCNT:
0132 8DD4          TEST     R13            ! TEST <CNT> !
0134 F643          JR       Z,EXERR        ! <CNT>=0, ERROR !

                ! MAIN LOOP FOR PRINTING DISPLAY LINES !
                NEWL:
0136 2109          LD       R9,#16         ! BYTES/LINE !
0138 0010
013A A1B5          LD       R5,R11
013C D032          CALR     CONVW          ! CONVERT BYTE !
013E 69E0          INC      OUTPTR(R14)
0140 0306
0142 76E8          LDA      R8,OUTBUF(R14) ! SET DISPLAY !
0144 0080
0146 0108          ADD      R8,#53
0148 0035                                  ! FORMAT !
014A 0C85          LDB      @R8,#'*'
014C 2A2A
014E A980          INC      R8

                ! LOOP FOR DISPLAY B/W/L UNITS !
```

```
                PUTONE:
0150 20BD         LDB       RL5,@R11      ! FETCH MEMORY !
0152 2E8D         LDB       @R8,RL5
0154 CA0D         CPB       RL5,#' '      ! CK FOR CHAR !
0156 2020
0158 E703         JR        C,DONTP       ! DONT PRINT !
015A CA0D         CPB       RL5,#%7F      ! CK FOR PRNT CHR !
015C 7F7F
015E E702         JR        C,RITE        ! YES, PRINT !
                DONTP:
0160 0C85         LDB       @R8,#'.'      ! REPLACE WITH '.' !
0162 2E2E
                RITE:
0164 A980         INC       R8,#1
0166 0C85         LDB        @R8,#'*'
0168 2A2A
016A D047         CALR      CONVB         ! CONVERT WORD !
016C A9B0         INC       R11,#1        ! INC MEM PTR !
016E AB90         DEC       R9
0170 FE08         JR        NZ,SPACE      ! NOT EOL !
0172 A980         INC       R8
0174 6FE8         LD        OUTPTR(R14),R8   ! SET PTR TO EO-BUF !
0176 0306
0178 D07E         CALR      PRNTBF        ! PRINT LINE !
017A A07F         LDB       RL7,RH7       ! RESET B/SP CNT !
017C ABD0         DEC       R13
017E EEDB         JR        NZ,NEWL    ! START NEW LINE !
0180 9E08         RET                     ! OR ELSE DONE !

                ! CHECK FOR POSITION OF SPACE !
                SPACE:
0182 FF1A         DBJNZ     RL7,PUTONE  ! NO SPACE YET !
0184 69E0         INC       OUTPTR(R14)       ! PUT SPACE !
0186 0306
0188 A07F         LDB       RL7,RH7     ! RESET B/SP CNT !
018A ABD0         DEC       R13
018C EEE1         JR        NZ,PUTONE   ! CONTINUE..... !
018E A980         INC       R8
0190 6FE8         LD        OUTPTR(R14),R8
0192 0306
0194 D08C         CALR      PRNTBF      ! FINISHED !
0196 9E08         RET

                ! SUBSTITUTION MODE, SINGLE  DISPLAY !
                SUBMOD:
0198 A1B9         LD        R9,R11
019A A1B5         LD        R5,R11
019C D062         CALR      CONVW         ! CONVERT BYTE !
019E 69E0         INC       OUTPTR(R14)
01A0 0306
```

```
                AGAIN:
01A2 20RD          LDB       RL5,@R11
01A4 D064          CALR      CONVB          ! CONVERT WORD !
01A6 A9P0          INC       R11,#1
01A8 FF04          DBJNZ     RL7,AGAIN
01AA 69E0          INC       OUTPTR(R14),#1     ! INSERT SPACE !
01AC 0306
01AE A07F          LDB       RL7,RH7        ! RESET B/SP CNT !

                ! GET SUBSTITUTION OR 'Q' OR CR !
01B0 D051          CALR      DISPNC
01B2 9E07          RET       C
01B4 E6F1          JR        Z,SUBMOD       ! GOT CR,.....!
01B6 2F93          LD        @R9,R3          ! SUBST. WORD !
01B8 A07F          LDB       RL7,RH7     ! RESET B/SP CNT !
01BA F8FE          JP        SUBMOD

                EXERR:
01BC 34CA          LDA       R10,R12(#EROR)
01BE 00C0*
01C0 1EA8          JP        @R10

01C2            END DISPLAY


                GLOBAL BIARG LABEL
01C2            GLOBAL TRIARG PROCEDURE
                !*****************************************
                 *  TRIARG: GETS NEXT THREE ARG AFTER    *
                 *        CMD, INTO INTBUF; ASCII-TO-    *
                 *        HEX CONVERSION IS PERFORMED    *
                 *        ON ADDRESSES.                  *
                 *                                       *
                 *  BIARG:  SEEKS NEXT TWO ARG IN THE    *
                 *       SAME MANNER.                    *
                 *                                       *
                 *  REG USE: RETURN  R3 = 3RD ARG (TRI)  *
                 *                   R4 = 2ND ARG        *
                 *                   R5 = 1ST ARG        *
                 *                                       *
                 *                   R3 = 2ND ARG (BI)   *
                 *                   R4 = 1ST ARG        *
                 *                   AND NC IF ALL ARG   *
                 *                                       *
                 *****************************************!

                ENTRY
01C2 D0C7          CALR      GETNXT       ! SKIP REST OF CMD !
01C4 D0DD          CALR      GETADR       ! GET 1ST ARG !
```

```
01C6 E608       JR      Z, ERRSTP     ! CR=ERROR !
01C8 A135       LD      R5,R3
01CA E801       JR      GET2ND
              ! TWO ARGUMENTS ENTRY POINT !
              BIARG:
01CC D0CC       CALR    GETNXT

              GET2ND:
01CE D0E2       CALR    GETADR        ! GET NEXT ARG !

01D0 E603       JR      Z,ERRSTP      ! CR=ERROR !
01D2 A134       LD      R4,R3
01D4 D0E5       CALR    GETADR        ! GET LAST ARG !

01D6 9E0F       RET     NC
              ERRSTP:
01D6 8D81       SETFLG  C             !SIGNAL ERROR!
01DA 9E08       RET
01DC          END TRIARG

              GLOBAL
01DC          MOVE  PROCEDURE
              !*****************************************
               *                                       *
               *  MOVE: MOVES DATA IN MEMORY.          *
               *                                       *
               *  REG USE: RETURN  R5 = <ADR>          *
               *                   R4 = <NEW ADR>      *
               *                                       *
               *****************************************!
              ENTRY
01DC D00E       CALR    TRIARG          ! GET THREE ARGS !
01DE E70D       JR      C,WTF         ! NOT ENOUGH ARG !
01E0 A156       LD      R6,R5
01E2 8346       SUB     R6,R4       ! FIND MOVE DIRECT !
01E4 E703       JR      C,UP        ! WILL MOVE UP.... !
01E6 BA51       LDIRB   @R4,@R5,R3  ! WILL MOVE DOWN !
01E8 0340
01EA 9E08       RET
              UP:
01EC 8135       ADD     R5,R3
01EE AB50       DEC     R5
01F0 8134       ADD     R4,R3
01F2 AB40       DEC     R4
01F4 BA59       LDDRB   @R4,@R5,R3  ! MOVE BLOCK !
01F6 0340
01F8 9E08       RET

              WTF:
01FA 34CA       LDA     R10,R12(#EROR)
```

```
01FC 0000*
01FE 1EA8      JP       @R10

0200         END MOVE

0200         GLOBAL FILL PROCEDURE
             !*****************************************
              *                                       *
              *  FILL: STROES GIVEN DATA WORD IN ALL  *
              *        SPECIFIED MEMORY LOCATIONS.    *
              *                                       *
              *****************************************!
             ENTRY
0200 34CA      LDA      R10,R12(#EROR)
0202 0000*
0204 D022      CALP      TRIARG        ! FETCH THREE ARGS !
0206 1EA7      JP        C,@R10
0208 A750      BIT       R5,#0         !MUST BE EVEN ADR!
020A 1EAE      JP        NZ,@R10
020C 8354      SUB       R4,R5         !COMPARE START TO!
                                           !END!
020E 1EA7      JP        C,@R10
0210 B341      SRL       R4,#1         !SET COUNT!
0212 FFFF
0214 2F53      LD        @R5,R3        !STORE DATA AT START!
0216 9E06      RET       Z             !ONLY ONE REQ !
0218 A153      LD        R3,R5
021A A931      INC       R3,#2         !NEXT MEM LOC !
021C BB51      LDIR      @R3,@R5,R4    ! FILL ALL !
021E 0430
0220 9E08      RET
0222         END FILL

             END DISPLAY_CMD
```

6. BRK_QUIT MODULE

```
Z8000ASM  2.02
LOC     OBJ CODE     STMT SOURCE STATEMENT

                       1 BRK_QUIT_CMD MODULE
                   $LISTON  $TTY

                   !*****************************************
                    *                                       *
                    *  BREAK: COMMAND TO RESET A BREAK ADR  *
                    *         REMOVES AN OLD ONE, AND SETS  *
                    *         A NEW ONE.                    *
                    *                                       *
                    *  QUIT: COMMAND TO ENTER THE TRANSPAR.*
                    *        MODE WHERE ALL RECEIVED CHAR   *
                    *        ARE ECHOED BETWEEN MCZ SYS     *
                    *        AND CONSOLE. Z8000 FUNCTIONS   *
                    *        IN RELAY CAPACITY ONLY.        *
                    *                                       *
                    *  SYNTAX:     BREAK  [<ADR>]           *
                    *              QUIT                     *
                    *                                       *
                    *****************************************!

                   CONSTANT

                   ( INCLUDE GLOBAL CONSTANTS )

                   EXTERNAL   GETNXT    PROCEDURE
                   EXTERNAL   GETADR    PROCEDURE
                   EXTERNAL   GMCZAD    PROCEDURE
                   EXTERNAL   SNDMCZ    PROCEDURE
                   EXTERNAL   GETBUF    PROCEDURE
                   EXTERNAL   SNDCHR    PROCEDURE
                   EXTERNAL   EROR      LABEL
                   EXTERNAL   SKPBLK    PROCEDURE
                   EXTERNAL   CONVERT   PROCEDURE
                   EXTERNAL   GETCHR    PROCEDURE


                   GLOBAL
                   $SECTION BRK_QUIT_PROC
                   $REL 0
```

```
                GLOBAL
0000            ASCHEX  PROCEDURE
                !*************************************
                 *                                   *
                 *  ASCHEX: ROUTINE TO CONVERT ONE LINE *
                 *          OF ASCII INPUT TO HEX VALUE *
                 *          (8-ASCII CHR TO ONE LWORD)  *
                 *                                   *
                 *  REG USE: RETURN  RR2 = 8-ASCII CHR  *
                 *                   AND C IF REC 'Q'   *
                 *                       NC,Z IF CR     *
                 *                       NC,NZ IF SPACE *
                 *                                   *
                 *************************************!
                ENTRY
0000 C82A          LDB      RL0,#'*'
0002 34CA          LDA      R10,R12(#SNDCHR)
0004 0000*
0006 1FA0          CALL     @R10            ! OUTPUT PROMPT !
0008 DFE0          CALR     GETLNE          ! GET LINE INTBUF !
000A 8D29          CLR      R2
000C 8D38          CLR      R3              ! CLR RETURN REGS !
000E 34CA          LDA      R10,R12(#SKPBLK)
0010 0000*
0012 1FA0          CALL     @R10
0014 9E06          RET      Z               ! REC CR !
0016 0A09          CPB      RL0,#'Q'
0018 5151
001A EE02          JR       NZ,GB3          ! NOT 'Q',.....!
001C 8D81          SETFLG   C
001E 9E08          RET                      ! RETURN WITH 'Q' !

                GB3:
0020 34CA          LDA      R10,R12(#CONVERT)
0022 0000*
0024 1FA0          CALL     @R10          ! ASCII TO HEX !
0026 E70E          JR       C,ENDERR      ! NON-ASCII CHR !
0028 BEB8          RLDB     RL0,RL3
002A BE38          RLDB     RL0,RH3
002C BEA8          RLDB     RL0,RL2
002E BE28          RLDB     RL0,RH2       ! ROTATE NIBBLE !
0030 34CA          LDA      R10,R12(#GETCHR)
0032 0000*
0034 1FA0          CALL     @R10     ! GET NEXT ASCII CHAR !
0036 E604          JR       Z,NOMORE   ! GOT NOTHING !
0038 0A08          CPB      RL0,#' '   ! END OF INPUT !
003A 2020
003C EEF1          JR       NZ,GB3     ! YES, ALL DONE !
003E 8D83          RESFLG   C
                NOMORE:
```

```
0040 8D43      RESFLG   Z
0042 9E08      RET

               ENDERR:
0044 34CA      LDA      R10,R12(#EROR)
0046 0000*
0048 1EA8      JP       @R10
004A           END ASCHEX

               GLOBAL
004A           GETLNE  PROCEDURE
               !*****************************************
                *                                       *
                *  GETLNE: REC ONE LINE INPUT FROM      *
                *          CONS (PORT2), UP TO 80-CHR   *
                *          MAX, STORE IN INTBUF PLUS    *
                *          CR, AND ECHO BACK TO CONS.   *
                *                                       *
                *  REG USE: RETURN  RL0= 1ST CHR IN BUF*
                *                   AND Z IF CHR = CR   *
                *                                       *
                *****************************************!
               ENTRY
004A 76E2      LDA       R2,INTBUF(R14)   ! GET BASE INTBUF !
004C 0000
004E 2101      LD        R1,#INSIZ    ! GET MAX SIZE !
0050 0080
0052 6FE2      LD        INTPTR(R14),R2
0054 0304
0056 DFF7      CALR      CONSOL       ! FILL LINE IN INTBUF!
0058 E6F5      JR        Z,ENDERR     ! LINE TOO LONG !
005A 34CA      LDA       R10,R12(#SKPBLK)
005C 0000*
005E 1FA0      CALL      @R10
0060 6BE0      DEC       INTPTR(R14)      ! RETURN TO START !
0062 0304
0064 0A08      CPB       RL0,#CR
0066 0D0D
0068 9E08      RET                        ! GOT 1ST CHR = CR !
006A           END GETLNE
```

```
                GLOBAL
006A            CONSOL  PROCEDURE
                !*****************************************
                 *                                       *
                 *  CONSOL: STORE CONS INPUT LINE IN     *
                 *          BUFFER ADDRESS PROVIDED,     *
                 *          PLACE CR AT END OF LINE,     *
                 *          AND PROVIDE DELETE CHAR      *
                 *          AND DELETE LINE EDIT FUNC.   *
                 *                                       *
                 *  REG USE: INPUT  R1 = SIZE OF BUFFER  *
                 *                  R2 = BUFFER ADR      *
                 *           RETURN R1 = # OF REC CHR    *
                 *                  AND Z IF BUF LIMIT   *
                 *                                       *
                 *****************************************!
                ENTRY
006A 6FE2          LD       BUFADR(R14),R2     ! SAVE BOTH BUF ADR !
006C 0300
006E 6FE1          LD       BUFSIZ(R14),R1     ! AND BUF SIZE !
0070 0302
                HDNG:
0072 61E2          LD       R2,BUFADR(R14)
0074 0300
0076 8D18          CLR      R1
                REDLOP:
0078 DFC1          CALR     CONRD      ! GET CHAR AND ECHO !
007A 0A08          CPB      RL0,#%61   ! CONVERT TO UPPER CASE!
007C 6161
007E E704          JR       C,UPCASE   ! NOT LOWER CASE !
0080 0A08          CPB      RL0,#%7B
0082 7B7B
0084 EF01          JR       NC,UPCASE  ! YES LOWER CASE !
0086 A285          RESB     RL0,#5     ! CONVERT TO UPCASE !
                UPCASE:
0088 2E28          LDB      @R2,RL2

                ! PERFORM EDIT FUNCTIONS ON INPUT !
008A 0A08          CPB      RL0,#BS    !CK FOR DEL CHR !
008C 0808
008E EE11          JR       NZ,CONTCK      ! NO, CONTINUE CK !
0090 AB20          DEC      R2,#1          ! YES, BACKSPACE !
0092 AB10          DEC      R1
0094 4BE2          CP       R2,BUFADR(R14)     ! NOT TOO FAR !
0096 0300
0098 E707          JR       C,DO_OVR
009A C820          LDB      RL0,#' '
009C 34CA          LDA      R10,R12(#SNDCHR)
009E 0000*
00A0 1FA0          CALL     @R10           ! BLANK OUT BAD CHR !
```

```
00A2 C808      LDB     RL0,#BS
00A4 1FA0      CALL    @R10
00A6 E8E8      JR      REDLOP      ! CONTINUE.....!

               DO_OVR:
00A8 C82A      LDB     RL0,#'*'
00AA 34CA      LDA     R10,R12(#SNDCHR)
00AC 0000*
00AE 1FA0      CALL    @R10        ! SEND PROMPT !
00B0 E8E0      JR      HDNG        ! START AGAIN !

               CONTCK:
00B2 0A08      CPB     RL0,#LINDEL   ! CK FOR LINE DEL !
00B4 7F7F
00B6 E609      JR      Z,DELINE    ! YES,.....!
00B8 A920      INC     R2,#1
00BA A910      INC     R1
00BC 0A08      CPB     RL0,#CR     ! CK FOR CR !
00BE 0D0D
00C0 E60E      JR      Z,ADDLF     ! YES, ADD LF CHR !
00C2 4BE1      CP      R1,BUFSIZ(R14)    ! SIZE CK !
00C4 0302
00C6 EED8      JR      NZ,REDLOP   ! OK, GET NEXT CHR !
00C8 9E06      RET     Z           ! TOO LARGE, ERROR !

               DELINE:
00CA C85E      LDB     RL0,#%5E
00CC 34CA      LDA     R10,R12(#SNDCHR) ! SND LINE DEL !
00CE 0000*
00D0 1FA0      CALL    @R10
00D2 DFF5      CALR    NEWLNE      ! START NEW LINE !
00D4 C820      LDB     RL0,#' '
00D6 34CA      LDA     R10,R12(#SNDCHR)
00D8 0000*
00DA 1FA0      CALL    @R10        ! SND CHR !
00DC E8CA      JR      HDNG        ! START AGAIN !

               ADDLF:
00DE C80A      LDB     RL0,#LF
00E0 34CA      LDA     R10,R12(#SNDCHR)
00E2 0000*
00E4 1FA0      CALL    @R10        ! SEND LF CHR !
00E6 8D43      RESFLG  Z
00E8 9E08      RET
00EA           END CONSOL
```

```
                GLOBAL
00EA            NEWLNE  PROCEDURE
                !*****************************************
                 *                                       *
                 *  NEWLNE: SENDS CR AND LF TO CONSOLE   *
                 *                                       *
                 *****************************************!
                ENTRY
00EA 34CA          LDA       R10,R12(#SNDCHR)
00EC 00C0*
00EE C80D          LDB       RL0,#CR
00F0 1FA0          CALL      @R10
00F2 C80A          LDB       RL0,#LF
00F4 1FA0          CALL      @R10       ! ADR SNDCHR IN R10 !
00F6 9FC8          RET
00F8            END NEWLNE

                GLOBAL
00F8            CONRD  PROCEDURE
                !*****************************************
                 *                                       *
                 *  CONRD: GETS CHAR FROM CONSOLE INPUT  *
                 *         BUFFER (INTBUF) AND ECHOS     *
                 *         BACK TO CONSOLE. LOOPS UNTIL  *
                 *         RECEIVE CHARACTER.            *
                 *                                       *
                 *  REG USE: RETURN  RL0= CHR            *
                 *                   AND Z IF CHR=CR     *
                 *                                       *
                 *****************************************!
                ENTRY
00F8 61F0       TC:LD         R0,GETOUT(R14)
00FA 030E
00FC 4BE0          CP         R0,NXTPTR(R14)     ! COMPARE GET AND !
00FE 030C
                                                 ! PUT PTRS !
0100 E6FB          JR         Z,TC               ! REC NOTHING.... !
0102 93F2          PUSH       @R15,R2
0104 34CA          LDA        R10,R12(#GETBUF)
0106 0000*
0108 1FA0          CALL       @R10           ! GET RNGBUF ADR !
010A 6FE0          LD         GETOUT(R14),R0
010C 030E
010E 2028          LDB        RL0,@R2        ! STO CHR FOR RTN !
0110 97F2          POP        R2,@R15
                ! CHECK FOR NON-DISPLAY FROM LOAD_FILE !
0112 0B09          CP         R9,#%AAAA
0114 AAAA
0116 E603          JR         Z,NO_DISPLAY
0118 34CA          LDA        R10,R12(#SNDCHR)
```

```
011A 0000*
011C 1FA0      CALL     @R10

               NO_DISPLAY:
011E 9E08      RET
0120         END CONRD

             GLOBAL
0120         BREAK  PROCEDURE
             !*****************************************
              *                                       *
              *  BREAK: CLEARS PPEVIOUSLY SET BREAK   *
              *         POINT BY REPLACING UNIMP.     *
              *         INSTRUCTION (0E00) WITH ORG   *
              *         INSTRUCTION FOR THE ADR, AND  *
              *         SETS A NEW BREAK POINT IF     *
              *         SPECIFIED (<ADR>).            *
              *                                       *
              *****************************************!
             ENTRY
             ! CLEAR PREVIOUS BREAK POINT !
0120 61E2      LD       R2,BRKADR(R14)
0122 0316
0124 61E1      LD       R1,BRKSTR(R14)
0126 0314
0128 2F21      LD       @R2,R1       ! RESTORE INST !
012A 4DE8      CLR      BRKADR(R14)
012C 0316
012E 4DE8      CLR      BRKSTR(R14)
0130 0314
0132 4DE5      LD       BRKCNT(R14),#1   ! RESET CNT TO 1 !
0134 030A
0136 0001
0138 34CA      LDA      R10,R12(#GETNXT)
013A 0000*
013C 1FA0      CALL     @R10      ! GET NEXT ARGUMENT !
013E 9E06      RET      Z         ! NO NEW ADDRESS !

             ! GET NEW BREAK ADDRESS !
0140 34CA      LDA      R10,R12(#GETADR)
0142 0000*
0144 1FA0      CALL     @R10        ! GET BRK ADR !
0146 A132      LD       R2,R3       ! SAVE BRK ADR !
0148 E607      JR       Z,GB4       ! NO BRKCNT...!

             ! GET NEW BREAK COUNT !
014A 34CA      LDA      R10,R12(#GETADR)
014C 0000*
014E 1FA0      CALL     @R10        ! GET COUNT !
0150 8D34      TEST     R3          ! TEST FOR ZERO !
```

```
0152 E602       JR      Z,GB4       ! BRKCNT = 1 !

                NCZ:
0154 6FE3       LD      BRKCNT(R14),R3  ! SET BRKCNT !
0156 030A

                ! SET UNIMPLEMENTED INSTRUCTION IN BREAK ADR !
                GB4:
0158 A320       RES     R2,#0       ! MAKE EVEN ADR !
015A 6FE2       LD      BRKADR(R14),R2
015C 0316
015F 2121       LD      R1,@R2
0160 6FE1       LD      BRKSTR(R14),R1  ! SAVE INST !
0162 0314
0164 61E1       LD      R1,UNIMP(R14)
0166 0308
0168 2F21       LD      @R2,R1      ! PLACE '0E00' !

                ! CHECK FOR WRITE TO EXISTING MEMORY !
016A 2121       LD      R1,@R2
016C 4BE1       CP      R1,UNIMP(R14)
016E 0308
0170 9E06       RET     Z
0172 4DE5       LD      BRKCNT(R14),#1      ! MEMORY NOT THERE !
0174 030A
0176 0001
0178 4DE8       CLR     BRKADR(R14)
017A 0316
017C 4DE8       CLR     BRKSTR(R14)
017E 0314
0180 34CA       LDA     R10,R12(#EROR)
0182 0000*
0184 1FA8       JP      @R10            ! GOT WRONG BRK PNT !

0186            END BREAK

                GLOBAL
0186            QUIT  PROCEDURE
                !*****************************************
                 *                                       *
                 *  QUIT: TRANSMITS ALL CHR AND CR FROM  *
                 *        CONS TO MCZ; THEN RELAYS ALL   *
                 *        TO CONS FROM MCZ; AND ETC.     *
                 *                                       *
                 *****************************************!
                ENTRY
0186 4DE8       CLR     MCZPUT(R14)
0188 0310
018A 4DE8       CLR     MCZGET(R14)
018C 0312
```

```
018E 65E0      SET       MFLAGS(R14),#TRPMDE  ! RESET BUF PTRS !
0190 031C
                                  ! AND ENTER TRANSPARENT MODE !

               ! CONSOLE RECEIVE ROUTINE !
               PORTB:
0192 61E0      LD        R0,GETOUT(R14)
0194 030E
0196 4BE0      CP        R0,NXTPTR(R14)   ! CK FOR CONS INPUT !
0198 030C
019A E60A      JR        Z,PORTA      ! NO, CK MCZ..... !

               ! PROCESS CONSOLE INPUT !
019C 34CA      LDA       R10,R12(#GETBUF)
019E 0000*
01A0 1FA0      CALL      @R10         ! GET RNGBUF ADR !
01A2 6FE0      LD        GETOUT(R14),R0   ! SET BEGIN PTR !
01A4 030E
01A6 2028      LDB       RL0,@R2
01A8 34CA      LDA       R10,R12(#SNDMCZ)
01AA 0000*
01AC 1FA0      CALL      @R10         ! ECHO CHR TO MCZ !
01AE EEF1      JR        NZ,PORTB     ! CONTINUE UNTIL CR !

               ! MCZ RECEIVE ROUTINE !
               PORTA:
01B0 61E0      LD        R0,MCZGET(R14)
01B2 0312
01B4 4BE0      CP        R0,MCZPUT(R14)     ! CK FOR MCZ INPUT !
01B6 0310
01B8 E6EC      JR        Z,PORTB       ! NO, CK CONSOLE...!
01BA 34CA      LDA       R10,R12(#GMCZAD)
01BC 0000*
01BE 1FA0      CALL      @R10          ! GET MCZBUF ADR !
01C0 6FE0      LD        MCZGET(R14),R0
01C2 0312
01C4 2028      LDB       RL0,@R2    ! GET CHAR FROM MCZBUF !
01C6 34CA      LDA       R10,R12(#SNDCHR)
01C8 0000*
01CA 1FA0      CALL      @R10      ! OUTPUT CHR TO CONSOLE !
01CC E8F1      JR        PORTA    ! CONTINUE TIL EMPTY !
01CE           END QUIT

               END BRK_QUIT_CMD
```

7. DEBUG MODULE

```
Z8000ASM  2.02
LOC     OBJ CODE     STMT SOURCE STATEMENT

                       1 DEBUG_CMD MODULE
                  $LISTON  $TTY

                  !*****************************************
                   *                                       *
                   *  JUMP CMD: CHANGES USER PC VALUE AND  *
                   *            BEGINS PROGRAM EXECUTION   *
                   *            AT THAT POINT <PC>.        *
                   *                                       *
                   *  GO CMD:   BEGINS PROGRAM EXECUTION   *
                   *            AT THE CURRENT USER PC.    *
                   *                                       *
                   *  NEXT CMD: STEPS THROUGH PROGRAM      *
                   *            EXECUTION ONE INST. AT A   *
                   *            TIME OR IN MULTIPLES <N>.  *
                   *                                       *
                   *  SYNTAX:       JUMP <PC>              *
                   *                GO                     *
                   *                NEXT [<N>]             *
                   *                                       *
                   *****************************************!

                  CONSTANT

                     RXR        :=  2
                     TXR        :=  0
                     PAR        :=  7
                     PORTAD     :=  %FFD9
                     PORTBD     :=  %FFE1
                     PORTAC     :=  %FFDB
                     PORTBC     :=  %FFE3

                     IDPORT     :=  %FFC3
                     ICPORT     :=  %FFC9

                     TCMD    :=  %FFD2
                     TDTA    :=  %FFD0

                     BUS_LOCK :=%FFF9
                     BUS_UNLOCK :=%FFFB
                     VINTR      :=  %(2)0001000000000000
                     VIBIT      :=  12
                     ESCAPE     :=  %1B
                     BS         :=  %08
```

```
                LINDEL    :=  %7F
                CR        :=  %0D
                LF        :=  %0A
                TXOFCH    :=  %13
                TXONCH    :=  %11
                INSIZ     :=  128        ! INTBUF SIZE !
                OUTSIZ    :=  128        ! OUTBUF SIZE !
                RBSIZ     :=  256        ! RING BUFFER SIZE !
              ! BIT POSITIONS IN MONITOR FLAG WORD !
                TRPMDE    :=  0
                ISTOP     :=  1
                OSTOP     :=  2
                SNDMDE    :=  3
                LDMDE     :=  4
                ESC       :=  5
                TXMSK     := %6


              INTERNAL
              $SECTION DATA_DEC
              $ABS 0

0000            INTBUF    ARRAY [128 BYTE]
0080            OUTBUF    ARRAY [128 BYTE]
0100            RNGBUF    ARRAY [256 BYTE]
0200            MCZBUF    ARRAY [256 BYTE]

0300            BUFADR    WORD
0302            BUFSIZ    WORD

0304            INTPTR     WORD
0306            OUTPTR    WORD

0308            UNIMP     WORD
030A            BRKCNT    WORD

030C            NXTPTR    WORD
030E            GETOUT    WORD
0310            MCZPUT    WORD
0312            MCZGET    WORD
0314            BRKSTR    WORD
0316            BRKADR    WORD
0318            TMPSP     WORD
031A            TMPFCW    WORD

031C            MFLAGS    WORD

               ! USER REGISTER STORAGE !

031E            R0_       WORD
```

```
0320          R1_       WORD
0322          R2_       WORD
0324          R3_       WORD
0326          R4_       WORD
0328          R5_       WORD
032A          R6_       WORD
032C          R7_       WORD
032E          R8_       WORD
0330          R9_       WORD
0332          R10_      WORD
0334          R11_      WORD
0336          R12_      WORD
0338          R13_      WORD
033A          R14_      WORD
033C          R15_      WORD
033E          RPC_      WORD
0340          RFC_      WORD

0342          RETRY     WORD
0344          ADR_STR   WORD

          INTERNAL
          $SECTION PSA_DATA
          $ABS 0
0000          PSA    RECORD [
                        DATA_AREA      WORD
                        CODE_AREA      WORD
                        UNIMP_INST     LONG
                        PRIV_INST      LONG
                        SYSTEM_CALL    LONG
                        SEG_TRAP       LONG
                        NMI_INT        LONG
                        NVI_INT        LONG
                        VEC_FCW        WORD
                        VEC_PC         ARRAY [200  WORD]
                        ]

          EXTERNAL
             GETNXT    PROCEDURE
             GETADR    PROCEDURE
             RGHDR1    PROCEDURE
             RGHDR2    PROCEDURE
             PRREG1    PROCEDURE
             PRREG2    PROCEDURE
             EROR      LABEL
             EXEC      PROCEDURE

          $SECTION DEBUG_PROC
          $REL 0
```

```
                GLOBAL
0000            DEBUG_ENTRY  PROCEDURE
                !*****************************************
                 *                                       *
                 *  DEBUG_ENTRY: RESTORES DMONITOR R12   *
                 *               AND R14 (CODE AND DATA)*
                 *               REGISTERS FOR INTERRUPT*
                 *               ENTRIES.                *
                 *                                       *
                 *****************************************!
                ENTRY
0000 93F1          PUSH     @R15,R1
0002 93FE          PUSH     @R15,R14
0004 7D15          LDCTL    R1,PSAPOFF     ! GET PSA BASE !
0006 211E          LD       R14,@R1        ! RESTORE DATA BASE !
0008 6FEC          LD       R12_(R14),R12  ! SAVE USER R12 !
000A 0336
000C 611C          LD       R12,CODE_AREA(R1)     ! CODE BASE !
000E 0002
0010 97F1          POP      R1,@R15
0012 6FE1          LD       R14_(R14),R1
0014 033A
0016 97F1          POP      R1,@R15
0018 9E08          RET

001A            END DEBUG_ENTRY


                GLOBAL
001A            SAVREG   PROCEDURE
                !*****************************************
                 *                                       *
                 *  SAVREG: SAVES USER PROGRAM STATUS    *
                 *          AND REGS 1-14 CONTENTS.      *
                 *                                       *
                 *****************************************!
                ENTRY
001A 6FEF          LD       TMPSP(R14),R15      ! RTN ADR !
001C 0318
001E 31F0          LD       R0,R15(#4)     !SAVE:  !
0020 0004
0022 6FE0          LD       RFC_(R14),R0         ! USER FCW !
0024 0340
0026 31F0          LD       R0,R15(#6)
0028 0006
002A 6FE0          LD       RPC_(R14),R0         ! USER PC !
002C 033E
                ! SAVE R1 - R14 !
002E 76EF          LDA      R15,R1_(R14)
0030 0320
```

```
0032 1CF9      LDM       @R15,R1,#11    ! STORE REGS !
0034 010A
0036 61EF      LD        R15,TMPSP(R14)     ! RESTORE SP !
0038 0318
003A 6FED      LD        R13_(R14),R13
003C 0338
003E 9E08      RET
0040         END SAVREG

             GLOBAL
0040         RESTOR  PROCEDURE
             !*****************************************
              *                                       *
              *  RESTOR: RESTORES USER PROGRAM STATUS*
              *          AND REGS 0-11,13 CONTENTS.   *
              *                                       *
              *****************************************!
             ENTRY
0040 6FEF      LD        TMPSP(R14),R15      ! SAVE STK PTR !
0042 0318
0044 61EF      LD        R15,R15_(R14)       ! SET USER SP !
0046 033C
0048 61E1      LD        R1,RFC_(R14)         ! RESTORE:    !
004A 0340
004C 33F1      LD        R15(#2),R1        ! USER FCW !
004E 0002
0050 61E1      LD        R1,RPC_(R14)
0052 033E
0054 33F1      LD        R15(#4),R1    ! USER PC !
0056 0004
             !RESTORE R0 - R15 !
0058 34EF      LDA        R15,R14(#R0_)
005A 031E
005C 1CF1      LDM       R0,@R15,#12    ! RESTORE REGS !
005E 0003
0060 61ED      LD        R13,R13_(R14)
0062 0338
0064 61EF      LD        R15,TMPSP(R14)      ! RESTORE SP !
0066 0318
0068 9E08      RET
006A         END RESTOR
```

```
                GLOBAL
                GO   LABEL
006A            JUMP PROCEDURE
                !****************************************
                 *                                      *
                 *  JUMP: TRANSFERS PROGRAM EXECUTION   *
                 *        TO LOCATION SPECIFIED IN CMD. *
                 *                                      *
                 *  GO:   BEGIN EXECUTION AT CURRENT    *
                 *        USER PC VALUE.                *
                 *                                      *
                 ****************************************!
                ENTRY
006A 34CA          LDA      R10,R12(#GETNXT)
006C 0000*
006E 1FA0          CALL     @R10           ! GET NEXT ARG !
0070 E616          JR       Z,SETERR       ! NO ARGS !
0072 34CA          LDA      R10,R12(#GETADR)
0074 0000*
0076 1FA0          CALL     @R10           ! RTN JMP ADR IN R3 !
0078 A330          RES      R3,#0          ! SET TO EVEN ADR !
007A 6FE3          LD       RPC_(R14),R3         ! SET NEW PC VALUE !
007C 033E
                !GLOBAL GO CMD !
007E 61E3       GO:LD       R3,RPC_(R14)
0080 033E
0082 4BE3          CP       R3,BRKADR(R14)      ! PC = BRKADR? !
0084 0316
0086 E665          JR       Z,BRKROU       ! YES TAKE INT. !
0088 D025          CALR     RESTOR         ! NO, RESTORE REGS !
008A 61EF          LD       R15,R15_(R14)       ! RESTORE SP !
008C 033C
008E 61EC          LD       R12,R12_(R14)
0090 0336
0092 93F1          PUSH     @R15,R1
0094 61E1          LD       R1,R14_(R14)
0096 033A
0098 A11E          LD       R14,R1
009A 97F1          POP      R1,@R15

009C 7B00          IRET                    ! BEGIN EXEC BY IRET !

                SETERR:
009E 34CA          LDA      R10,R12(#EROR)
00A0 0000*
00A2 1EA8          JP       @R10           ! RETURN TO EXEC !

00A4            END JUMP
```

```
                GLOBAL
00A4            BRKIHD  PROCEDURE
                !*****************************************
                 *                                       *
                 *  BRKIHD: BREAK INTERRUPT HANDLER      *
                 *          WHICH EXECUTES SINGLE USER   *
                 *          INST, PUTS BACK BREAK INST   *
                 *          (UNIMPLEMENTED INST.), AND   *
                 *          RESTORES USER FCW.           *
                 *                                       *
                 *****************************************!
                ENTRY
00A4 93F2          PUSH     @R15,R2
00A6 93F1          PUSH     @R15,R1        ! SAVE SYS REG 1-2 !
00A8 93FE          PUSH     @R15,R14
00AA 7D15          LDCTL    R1,PSAPOFF
00AC 211E          LD       R14,@R1        ! DATA_AREA ADR !
00AE 61E2          LD       R2,BRKADR(R14)
00B0 0316
00B2 61E1          LD       R1,UNIMP(R14)
00B4 0308
00B6 2F21          LD       @R2,R1         ! STORE UNIMP(R14) INST !
                                           ! FOR BREAK SIGNAL !
00B8 67EC          BIT      TMPFCW(R14),#VIBIT !CK VI EN !
00BA 031A
00BC EE05          JR       NZ,ALTHRU      ! YES, FINISHED !

00BE 31F2          LD       R2,R15(#8)     ! NO,.... !
00C0 0008
00C2 A32C          RES      R2,#VIBIT      ! ..SC EN VI !
00C4 33F2          LD       R15(#8),R2     ! AND PUT BACK !
00C6 0008

                ALTHRU:

                 ! RESTORE USER R12 AND R14 !
00C8 97FE          POP      R14,@R15
00CA 97F1          POP      R1,@R15
00CC 97F2          POP      R2,@R15        ! RESTORE REGS !
00CE 7B00          IRET
00D0            END BRKIHD
```

```
                GLOBAL
00D0            SSINT  PROCEDURE
                !*****************************************
                 *                                       *
                 *  SSINT: SINGLE STEP INT HANDLER FOR   *
                 *         USE WITH NEXT CMD. ALLOWS     *
                 *         SINGLE INST EXECUTION.        *
                 *                                       *
                 *****************************************!
                ENTRY
00D0 D069          CALR     DEBUG_ENTRY
00D2 6FEF          LD       R15_(R14),R15
00D4 033C
00D6 6FE0          LD       R0_(R14),R0
00D8 031E
00DA D061          CALR     SAVREG       ! SAVE ALL REGS !
00DC 61E2          LD       R2,BRKADR(R14)
00DE 0316
00E0 61E1          LD       R1,UNIMP(R14)
00E2 0308
00E4 2F21          LD       @R2,R1       ! RESTORE UNIMP(R14) INST !
00E6 67EC          BIT      TMPFCW(R14),#VIBIT
00E8 031A
00EA EE02          JR       NZ,DWN       ! VI EN !
00EC 63EC          RES      RFC_(R14),#VIBIT
00EE 0340

                DWN:
00F0 34CA          LDA      R10,R12(#PRREG1)
00F2 0000*
00F4 1FA0          CALL     @R10           ! OUTPUT REG CONTENT!
00F6 34CA          LDA      R10,R12(#PRREG2)
00F8 0000*
00FA 1FA0          CALL     @R10
00FC 6BE0          DEC      BRKCNT(R14),#1
00FE 030A
0100 EE1B          JR       NZ,SINST       ! CONTINUE SS !
0102 4D05          LD       BRKCNT,#1      ! RESET NEXT CNT !
0104 030A
0106 0001
0108 4DE8          CLR      MFLAGS(R14)
010A 031C
010C 7C05          EI       VI
010E 34CA          LDA      R10,R12(#EXEC)
0110 0000*
0112 1EA8          JP       @R10
0114            END SSINT
```

```
                GLOBAL
                BRKROU    LABEL
                SINST     LABEL
0114            NEXT      PROCEDURE
                !*****************************************
                 *                                       *
                 *  NEXT: SINGLE OR MULTIPLE STEPPING    *
                 *        THRU USER PROGRAM WITH REG     *
                 *        CONTENTS DISPLAYED AFTER EACH  *
                 *        INSTRUCTION EXECUTION.         *
                 *                                       *
                 *****************************************!
                ENTRY
0114 34CA          LDA       R10,R12(#GETNXT)
0116 0000*
0118 1FA0          CALL      @R10          ! SKIP REST CMD !
011A E605          JR        Z,KA          ! NO NEXT COUNT !
011C 34CA          LDA       R10,R12(#GETADR)
011E 0000*
0120 1FA0          CALL      @R10          ! GET COUNT !
0122 9D34          TEST      R3
0124 EE01          JR        NZ,KB         ! COUNT<0 !
0126 BD31       KA:LDK       R3,#1         ! SET CNT=1 !
0128 6FE3       KB:LD        BRKCNT(R14),R3    ! SET BRKCNT !
012A 030A
012C 34CA          LDA       R10,R12(#RGHDR1)
012E 0000*
0130 1FA0          CALL      @R10          ! DISPLAY HDR !
0132 34CA          LDA       R10,R12(#RGHDR2)
0134 0000*
0136 1FA0          CALL      @R10

                ! SINGLE INSTRUCTION EXECUTION ROUTINE !
                SINST:
0138 3402          LDAR        R2,SSINT     ! SETUP INT HDLR !
013A FF94
013C 61E3          LD        R3,BRKADR(R14)
013E 0316
0140 4FE3          CP        R3,RPC_(R14)       ! PC=BRKADR(R14)? !
0142 033E
0144 EE0D          JR        NZ,KC        ! NO CONTINUE....!
0146 61E3          LD        R3,BRKADR(R14)
0148 0316
014A 61E1          LD        R1,BRKSTR(R14)
014C 0314
014E 2F31          LD        @R3,R1       ! RESTORE ORIGINAL INST !
0150 E807          JR        KC           ! CONTINUE........!
                ! BREAK INTERRUPT HANDLER ROUTINE ENTRY !
                BRKROU:
0152 3402          LDAR        R2,BRKIHD    ! SET INT HDLR !
```

```
0154 FF4E
0156 61E3       LD      R3,BRKADR(R14)
0158 0316
015A 6101       LD      R1,BRKSTR   ! RESTORE BRK INST !
015C 0314
015E 2F31       LD      @R3,R1

              ! EXECUTE SINGLE USER INST AND RETURN BY IRET !
              KC:
0160 7C01       DI      VI
0162 7D15       LDCTL   R1,PSAPOFF
0164 7613       LDA      R3,VEC_PC(R1)
0166 001E
0168 3332       LD      R3(#12),P2      ! SET INT HDLR IN PSA !
016A 000C

016C D097       CALR    RESTOR      ! RESTORE REGS,PC,FCW !
016E 61EF       LD      R15,R15_(R14)    ! RESTORE SP !
0170 033C
0172 31F1       LD      R1,R15(#2)
0174 0002
0176 6FE1       LD      TMPFCW(R14),R1
0178 031A
017A 0501       OR      R1,#VINTR    ! ENABLE VI !
017C 1000
017E 33F1       LD      R15(#2),R1  ! PUT BACK !
0180 0002

               ! RESTORE USER R12 AND R14 !
0182 61E1       LD      R1,P1_(R14)
0184 0320
0186 93F1       PUSH    @R15,R1
0188 61EC       LD      R12,R12_(R14)
018A 0336
018C 61E1       LD      R1,R14_(R14)
018E 033A
0190 A11E       LD      R14,R1

0192 97F1       POP     R1,@R15
0194 3B16       OUT     %FFC0,R1    ! ARM SINGLE STEP !
0196 FFC0
0198 A1AA       LD      R10,R10
019A 7B00       IRET            ! RTN AND EXECUTE ONE INST !
019C          END NEXT
              END DEBUG_CMD
```

8. SEND MODULE

```
Z8000ASM  2.02
LOC     OBJ CODE     STMT SOURCE STATEMENT

                        1 SEND_CMD MODULE
                $LISTON  $TTY

                !*****************************************
                 *    SEND CMD:  SENDS SPECIFIC BLOCK OF *
                 *               MEM <BGN ADR><END ADR> TO  *
                 *               MCZ TO STORE AS <FNAME>    *
                 *               W/RELOAD OPT. ENTRY ADR    *
                 *               <ENT ADR>. ALL HEX VALUES  *
                 *               CONVERTED TO ASCII FOR     *
                 *               TRANS. IN TEKTRONIX FROMAT *
                 *               PACKETS; AND UP TO TEN     *
                 *               ATTEMPTS AT PACKET TRANS.  *
                 *               WILL BE MADE.              *
                 *                                          *
                 *    SYNTAX: SEND <BGNADR><ENDADR>         *
                 *                                [<ENT ADR>]*
                 *                                          *
                 *****************************************!

                CONSTANT

                   ( INCLUDE GLOBAL CONSTANTS )

                EXTERNAL
                    FNAME           PROCEDURE
                    BIARG           LABEL
                    GETADR          PROCEDURE
                    SNDMSG          PROCEDURE
                    SKIPLN          PROCEDURE
                    CMDPAS          PROCEDURE
                    CONVB           LABEL
                    CONVW           PROCEDURE
                    EROR            LABEL
                    GETACK          PROCEDURE
                    OUTLNE          PROCEDURE


                GLOBAL
                $SECTION SEND_PROC
                $REL 0
```

```
                GLOBAL
0000            SNDPAC PROCEDURE
                !*****************************************
                 *    SNDPAC: FORMATS TRANSFER PACKET    *
                 *            DATA IN TEXTRONIX FROMAT   *
                 *            AND SENDS TO MCZ.          *
                 *                                       *
                 *    REG USE: INPUT  R8 = BGN ADR       *
                 *                    R9 = WORD CNT      *
                 *             RETURN R8 = BGN ADDR OF   *
                 *                    NEXT AND Z IF LAST*
                 *                                       *
                 *****************************************!
                ENTRY
0000 8D38          CLR       R3
0002 C82F          LDB       RL0,#'/'
0004 6EE8          LDB       OUTBUF(R14),RL0     ! STORE / !
0006 0080
0008 69E0          INC       OUTPTR(R14),#1
000A 0306
000C 8D94          TEST      R9
000E 9E06          RET       Z

                !CONVERT BGN ADR TO ASCII, ADD TO CKSUM  !
0010 A185          LD        R5,R8
0012 34CA          LDA       R10,R12(#CONVW)
0014 0000*
0016 1FA0          CALL      @R10              ! CONVERT ASCII !
0018 A195          LD        R5,R9
001A 34CA          LDA       R10,R12(#CONVB)
001C 0000*
001E 1FA0          CALL      @R10              ! CONVERT WORD !
0020 A0BD          LDB       RL5,RL3
0022 34CA          LDA       R10,R12(#CONVB)
0024 0000*
0026 1FA0          CALL      @R10
                ! FORMAT ALL DATA !
0028 8D38          CLR       R3
                PROCLP:
002A 208D          LDB       RL5,@R8
002C 34CA          LDA       R10,R12(#CONVB)
002E 0000*
0030 1FA0          CALL      @R10
0032 A980          INC       R8,#1
0034 AB90          DEC       R9
0036 EEF9          JR        NZ,PROCLP     ! CONT CONVERSION !
0038 A0BD          LDB       RL5,RL3       ! DONE, STR CKSUM !
003A 34CA          LDA       R10,R12(#CONVB)
```

```
003C 0000*
003E 1FA0      CALL     @R10          ! CONVERT CKSUM !
0040 1F20      CALL     @R2
0042 8D43      RESFLG   Z
0044 9E08      RET
0046           END SNDPAC

               GLOBAL
0046           LASTPC PROCEDURE
               !*****************************************
                *   LASTPC:  FORMATS LAST PACKET FOR    *
                *            TRANS  WITH ENTRY ADR,     *
                *            COUNT=0, AND CKSUM.        *
                *                                       *
                *   REG USE:  INPUT   R6 = ENTRY ADR    *
                *                                       *
                *****************************************!
               ENTRY
0046 A165      LD       R5,R6
0048 34CA      LDA      R10,R12(#CONVW)
004A 0000*
004C 1FA0      CALL     @R10          ! CONVERT BYTE !
004E CD00      LDB      RL5,#0
0050 34CA      LDA      R10,R12(#CONVB)
0052 0000*
0054 1FA0      CALL     @R10          ! CONVERT WORD !
0056 A0BD      LDB      RL5,RL3       ! LOAD CKSUM !
0058 34CA      LDA      R10,R12(#CONVB)
005A 0000*
005C 1FA0      CALL     @R10
005E 1F20      CALL     @R2           ! SND TO....!
0060 9E08      RET
0062           END LASTPC

               GLOBAL
0062           ERMSG PROCEDURE
               !*****************************************
                *  ERMSG: SENDS ASCII // MSG TO         *
                *         MCZ TO ABORT SEND CMD.        *
                *****************************************!
               ENTRY
0062 C02F      LDB      RH0,#'/'
0064 C92F      LDB      RL0,#'/'
0066 6FE0      LD       OUTBUF(R14),R0    ! LOAD BUF // !
0068 0080
006A 76ED      LDA      R13,OUTBUF(R14)
006C 0080
006E A9D1      INC      R13,#2
0070 6FED      LD       OUTPTR(R14),R13
0072 0396
```

```
0074 34CA     LDA       R10,R12(#OUTLNE)
0076 0000*
0078 1FA0     CALL      @R10          ! OUTPUT LINE - MCZ !
007A 9E08     RET
007C          END ERMSG

              GLOBAL
007C          SEND PROCEDURE
              !*****************************************
               *                                       *
               *  SEND:  FORMATS AND TRANSFERS PACS    *
               *         CONTAINING DATA FROM THE      *
               *  SPECIFIED MEMORY AREA TO BE SAVED    *
               *  AS A MCZ FILE.                       *
               *                                       *
               *****************************************!
              ENTRY
007C 34CA     LDA       R10,R12(#FNAME)
007E 0000*
0080 1FA0     CALL      @R10          ! CK FNAME !
0082 0B0B     CP        R11,#%FFFE    ! CK FOR NO ADR !
0084 FFFE
0086 E60B     JR        Z,GOTERR
0088 A1B8     LD        R8,R11        ! SAVE BGN ADR !
008A 34CA     LDA       R10,R12(#GETADR)
008C 0000*
008E 1FA0     CALL      @R10
0090 E706     JR        C,GOTERR      ! ERROR !
0092 A137     LD        R7,R3         ! SAVE END ADR !
0094 34CA     LDA       R10,R12(#GETADR)
0096 0000*
0098 1FA0     CALL      @R10          ! GET ENT ADR !
009A A136     LD        R6,R3
009C 8387     SUB       R7,R8         ! # BYTES !
              GOTERR:
009E 34CA     LDA       R10,R12(#EROR)
00A0 0000*
00A2 1EA7     JP        C,@R10        ! RTN EXEC !
00A4 A970     INC       R7            ! BYTE COUNT !
00A6 65E3     SET       MFLAGS(R14),#SNDMDE
00A8 031C                             ! SIG SND MODE !
              ! SEND CMD TO MCZ TO LOAD SEND PROGRAM !
00AA 34CA     LDA       R10,R12(#CMDPAS)
00AC 0000*
00AE 1FA0     CALL      @R10          ! SND CMD !
00B0 9E06     RET       Z
00B2 34CA     LDA       R10,R12(#GETACK)  ! GET ACK !
00B4 0000*
00B6 1FA0     CALL      @R10
```

```
00B8 E603       JR      Z,AC
00BA 3402       LDAR    R2,OPERR        ! SEND FILE OPEN !
00BC 00B0
00BE F83B       JR      PEM             ! ERROR TO CONS !

00C0 67E5   AC:BIT      MFLAGS(R14),#ESC
00C2 031C
00C4 E607       JR      Z,AD            ! CK FOR ESC KEY !
                !SEND ABORT TO CONSOLE AND MCZ IF REC ESC KEY !
                SNDABT:
00C6 D033       CALR    ERMSG
00C8 3402       LDAR    R2,ABTMSG       ! SND ABORT MSG !
00CA 0086
00CC 34CA       LDA     R10,R12(#SNDMSG)
00CE 0000*
00D0 1FA0       CALL    @R10
00D2 9E08       RET

00D4 4DF5   AD:LD       RETRY(R14),#10
00D6 0342
00D8 000A
00DA 0B07       CP      R7,#30
00DC 001E
00DE EF03       JR      NC,COMP         ! #BYTES SND > 30 !
00E0 A179       LD      R9,R7           ! #<30, USE REST !
00E2 8D78       CLR     R7              ! LAST PACKET !
00E4 F804       JR      AE

                COMP:
00E6 2109       LD      R9,#30          ! SND 30-BYTES !
00E8 001E
00EA 0307       SUB     R7,#30          ! GET NEW #BYTES !
00EC 001E
00EE A19B   AE:LD       R11,R9          ! SAVE COUNT !

                ! MAIN RETRY LOOP FOR RESENDING PACKETS TO MCZ !
                RESEND:
00F0 67E5       BIT     MFLAGS(R14),#ESC  ! CK FOR USER ABORT !
00F2 031C
00F4 EEE8       JR      NZ,SNDABT       ! GOT, SND ABORT...!
00F6 34C2       LDA     R2,R12(#OUTLNE)
00F8 0000*
00FA D07E       CALR    SNDPAC
00FC E60F       JR      Z,LONE          ! SND LAST PACKET !
00FE 34CA       LDA     R10,R12(#SKIPLN)
0100 0000*
0102 1FA0       CALL    @R10            ! SKIP MCZ INPUT !
0104 34CA       LDA     R10,R12(#GETACK)
0106 0000*
0108 1FA0       CALL    @R10            ! WAIT FOR ACK !
```

```
010A EE01       JR      NZ,AF           ! REC NON-ACK !
010C E853       JR      AD              ! REC ACK !

010E E711   AF:JR       C,AG            ! REC 9 FOR ABORT !
0110 83B8       SUB     R8,R11
0112 A1B9       LD      R9,R11          ! RESEND !
0114 6BE0       DEC     RETRY(R14),#1
0116 0342
0118 E615       JR      Z,SNDSTP        ! SEND DONE !
011A E8EA       JR      RESEND

            ! SEND LAST PACKET TO MCZ !
            LONE:
011C 34C2       LDA      R2,R12(#OUTLNE)
011E 0000*
0120 D06E       CALR    LASTPC          ! PREPARE LAST PAC !
0122 34CA       LDA     R10,R12(#SKIPLN)
0124 0000*
0126 1FA0       CALL    @R10            ! SKIP MCZ ECHO !
0128 34CA       LDA     R10,R12(#GETACK)
012A 0000*
012C 1FA0       CALL    @R10            ! WAIT FOR ACK !
012E 9E06       RET     Z               ! FINISHED...!
0130 EF06       JR      NC,AZ           ! RESEND.....!

0132 3402   AG:LDAR      R2,WRTERR
0134 0024
            PEM:
0136 34CA       LDA     R10,R12(#SNDMSG)
0138 0000*
013A 1FA0       CALL    @R10            ! SEND WRITE ERROR !
013C 9E08       RET

013E 6BE0   AZ:DEC      RETRY(R14),#1
0140 0342
0142 EED6       JR      NZ,RESEND       ! TRY AGAIN !

            SNDSTP:
0144 D072       CALR    ERMSG
0146 3402       LDAR    R2,CKMSG
0148 0036
014A 34CA       LDA     R10,R12(#SNDMSG)
014C 0000*
014E 1FA0       CALL    @R10        ! SND CKSUM ERROR !
0150 9E08       RET
0152        END SEND
```

```
                ABTMSG:
0152 07            BVAL   %07
0153 2F            BVAL   '/'
0154 4142          WVAL   'AB'
0156 4F52          WVAL   'OR'
0158 54            BVAL   'T'
0159 0D            BVAL   %0D

                WRTERR:
015A 18            BVAL   %18
015B 2F            BVAL   '/'
015C 4649          WVAL   'FI'
015E 4C45          WVAL   'LE'
0160 2057          WVAL   ' W'
0162 5249          WVAL   'RI'
0164 5445          WVAL   'TE'
0166 2045          WVAL   ' E'
0168 5252          WVAL   'RR'
016A 4F52          WVAL   'OR'
016C 0D            BVAL   %0D
016D 20            BVAL   ' '

                OPERR:
016E 17            BVAL   %17
016F 2F            BVAL   '/'
0170 4F50          WVAL   'OP'
0172 454E          WVAL   'EN'
0174 2046          WVAL   ' F'
0176 494C          WVAL   'IL'
0178 4520          WVAL   'E '
017A 4552          WVAL   'ER'
017C 524F          WVAL   'RO'
017E 52            BVAL   'R'
017F 0D            BVAL   %0D

                CKMSG:
0180 12            BVAL   %12
0181 43            BVAL   'C'
0182 4B53          WVAL   'KS'
0184 554D          WVAL   'UM'
0186 2045          WVAL   ' E'
0188 5252          WVAL   'RR'
018A 4F52          WVAL   'OR'
018C 0D            BVAL   %0D
018D 20            BVAL   ' '

                END SEND_CMD

   0 errors
Assembly complete
```

APPENDIX C - BOOTLOAD Program Listing

A. PROM PROGRAMMING

The Bootload program listing contained in sections B - F was programmed into the firmware by the use of the two support programs (Z8XFER and CPMXFR) of Appendix E and the following procedures.

1. Bootup the SASS hardware to the monitor program by the following sequence of actions:

   RESET

   TYPE <CR>

   RESET MCZ System

   TYPE <CR>

   DEPRESS "INTA" switch (NMI).
   (displays: 'LOADING.....')

   When prompt appears ('*'), the monitor is loaded (approximately 20 sec).

2. Bootup the INTEL MDS System with a CP/M disk having the CPMXFR program.

3. On the SASS Developmental System, load the desired bootload program into memory at address 6000 HEX by the following command:

   TYPE "LOAD <Filename> 6000"

4. Load Z8000 transfer program (CPMXFR) by:

   TYPE "LOAD CPMXFR"

5. Setup to execute transfer program with the following commands:

   TYPE "F RPC <CR>"

   TYPE 'A900 <CR>'

   TYPE "5000 <CR>"

6. Connect the cable between the TTY port on the MDS system and connector 'A' on the SASS system. (connector 'A' is the MCZ port cable of MBC)

7. Execute the CP/M transfer program (CPMXFR) on the MDS system, selecting code transfer ('C').

8. On the SASS system, begin transfer by the following actions:

   TYPE "G <CR>"

After the transfer is completed, the SASS monitor will display the prompt ('*'), and the MDS system will provide an indication that two files (PROM08.OBJ and PROM09.OBJ) have been created.

To move the file first to ISIS-II and then to the PROM programmer, the following procedures apply:

9. On the MDS system with an ISIS-II disk in drive B:, transfer the two CP/M files to ISIS as follows:

   TYPE 'TOISIS PROM08.OBJ <CR>'

   TYPE "TOISIS PROM09.OBJ <CR>"

10. Reboot MDS system with ISIS-II disk in drive A: and and convert OBJ files to HEX files as follows:

    TYPE "OBJHEX PROM08.OBJ TO PROM08.HEX <CR>"

    TYPE "OBJHEX PROM09.OBJ TO PROM09.HEX <CR>"

11. Execute UPM (Universal PROM Programmer) by:

    PLACE CLEAN PROM IN PROGRAMMER

    TYPE "UPM <CR>"

    TYPE "READ FILE PROM08.HEX INTO 0000"

    TYPE 'PROGRAM FROM 0001 TO 2048 START 0000'

    PLACE CLEAN PROM IN PROGRAMMER (high byte)

    TYPE 'READ FILE PROM09.HEX INTO 0000'

    TYPE "PROGRAM FROM 0000 TO 2047 START 0000"

The PROM's are now programmed.

B. BOOTLOAD PROGRAM LISTING

1. BOOTLOAD1 MODULE

```
Z8000ASM  2.02
LOC     OBJ CODE     STMT SOURCE STATEMENT

                       1 BOOTLOAD1 MODULE
                  $LISTON  $TTY

                  CONSTANT

                     RXR         :=  2
                     TXR         :=  0
                     PAR         :=  7
                     PORTAD      :=  %FFD9
                     PORTBD      :=  %FFE1
                     PORTAC      :=  %FFDB
                     PORTBC      :=  %FFE3

                     IDPORT      :=  %FFCB
                     ICPORT      :=  %FFC9

                     ROM_DISABLE := %FFF0
                     TCMD    :=  %FFD2
                     TDTA    :=  %FFD0

                     BUS_LOCK :=%FFF9
                     BUS_UNLOCK :=%FFF9
                     VINTR       :=  %(2)0001000000000000
                     VIBIT       :=  12
                     ESCAPE      :=  %1B
                     BS          :=  %08
                     LINDEL      :=  %7F
                     CR          :=  %0D
                     LF          :=  %0A
                     TXOFCH      :=  %13
                     TXONCH      :=  %11
                     INSIZ       :=  128          ! INTBUF SIZE !
                     OUTSIZ      :=  128          ! OUTBUF SIZE !
                     RBSIZ       :=  256          ! RING BUFFER SIZE !
                  ! BIT POSITIONS IN MONITOR FLAG WORD !
                     TRPMDE      :=  0
                     ISTOP       :=  1
                     OSTOP       :=  2
```

```
                SNDMDE    :=  3
                LDMDE     :=  4
                ESC       :=  5
                TXMSK     := %6

                COMDS     :=  11
                MAX_CPU   :=  8

             TYPE
                MESSAGE    ARRAY[3 WORD]
                MEM_ARRAY   ARRAY[32 WORD]
                SWITCH     ARRAY[3 WORD]
                CPU_ENTRY        RECORD [
                          SIGNAL          WORD
                          CPU_ID          WORD
                          MSG_BLK         MESSAGE
                          MEM_MAP         MEM_ARRAY]
                ID_ARRAY              ARRAY[MAX_CPU WORD]
                ENTRY_ARRAY           ARRAY[MAX_CPU CPU_ENTRY]

             INTERNAL
             $SECTION TABLE1_DATA
             $ABS Ø

2000            CONFIG_TABLE    RECORD [
                          RW_PATTERN      WORD
                          CPU_NUM         WORD
                          NORM_RW_PAT     WORD
                          NORM_CPU_CNT    WORD
                          TABLE_LOCK      WORD
                          CPU_CNT         WORD
                          CPU_LIST        ENTRY_ARRAY]



             INTERNAL
             $SECTION DATA_DEC
             $ABS Ø
0000            INTBUF    ARRAY [128 BYTE]
0080            OUTBUF    ARRAY [128 BYTE]
0100            RNGBUF    ARRAY [256 BYTE]
0200            MCZBUF    ARRAY [256 BYTE]

0300            BUFADR    WORD
0302            BUFSIZ    WORD

0304            INTPTR     WORD
0306            OUTPTR    WORD

0308            UNIMP     WORD
```

```
030A            BRKCNT    WORD

030C            NXTPTR    WORD
030E            GETOUT    WORD
0310            MCZPUT    WORD
0312            MCZGET    WORD
0314            BRKSTR    WORD
0316            BRKADR    WORD
0318            TMPSP     WORD
031A            TMPFCW    WORD

031C            MFLAGS    WORD

              ! USER REGISTER STORAGE !

031E            R0_       WORD
0320            R1_       WORD
0322            R2_       WORD
0324            R3_       WORD
0326            R4_       WORD
0328            R5_       WORD
032A            R6_       WORD
032C            R7_       WORD
032E            R8_       WORD
0330            R9_       WORD
0332            R10_      WORD
0334            R11_      WORD
0336            R12_      WORD
0338            R13_      WORD
033A            R14_      WORD
033C            R15_      WORD
033E            RPC_      WORD
0340            RFC_      WORD

0342            RETRY     WORD
0344            ADR_STR   WORD
0346            CMDTBL    ARRAY [12 WORD]

            INTERNAL
            $SECTION PSA_DATA
            $ABS 0
0000            PSA   RECORD [
                        DATA_AREA      WORD
                        CODE_AREA        WORD
                        UNIMP_INST     SWITCH
                        PRIV_INST      SWITCH
                        SYSTEM_CALL    SWITCH
                        SEG_TRAP       SWITCH
                        NMI_INT        SWITCH
                        NVI_INT        SWITCH
```

```
                        VEC_FCW         WORD
                        VEC_PC          ARRAY [200  WORD]
                        ]

                $SECTION EXEC_PROC
                EXTERNAL
                   NMI     PROCEDURE
                   CONINT PROCEDURE
                   MCZHND PROCEDURE
                   BOOTLOAD_CPU  PROCEDURE
                   MEM_CPU       LABEL

                GLOBAL ERCR  LABEL
                GLOBAL EXEC  LABEL
0000            GLOBAL INITL PROCEDURE
                !*****************************************
                 *     DMONITOR INITIALIZATION           *
                 *****************************************!


                ENTRY
                ORGADR:
0000 0E00          WVAL     %0E00      ! UNIMP INST !
0002 4000          WVAL     %4000
0004 0008'         WVAL     STARTP
                PHYS_ID:
0006 F1F1          WVAL     %F1F1       ! UNIQUE PHYS_ID !

                !*****************************************
                 *                                       *
                 *       INDEPENDENT PROCESSOR STAGE     *
                 *                                       *
                 *****************************************!

                ! START OF INITIAL ENTRY TO DMONITOR !
                STARTP:
                !*****************************************
                 *                                       *
                 *  CLEAR_MEM: 'CLEARS' MEMORY BY WRIT-  *
                 *        ING R/W PATTERN <55AA> AT THE  *
                 *        BEGINING OF EACH BLOCK AND     *
                 *        CLEARING THE NEXT THREE WORD   *
                 *        LOCATIONS. CPU WILL WAIT AT    *
                 *        THE END FOR A PERIOD OF TIME   *
                 *        (APPROX. 2MSEC).               *
                 *                                       *
                 *****************************************!
0008 2104          LD       R4,#%F800      ! TOP MEM BLOCK !
000A F800
000C 2101          LD       R1,#%55AA      ! R/W PATTERN !
000E 55AA
```

```
0010 8D28      CLR     R2
0012 8D43      RESFLG  Z

               ! MAIN LOOP FOR CLEARING MEMORY !
               CLEAR_MEM:
                  DO
0014 A143            LD    R3,R4
0016 A931            INC   R3,#2
0018 2F41            LD    @R4,R1          ! STORE PATTERN !
001A 2105            LD    R5,#5
001C 0005
                     DO
001E 2F32               LD   @R3,R2
0020 A931               INC  R3,#2
0022 AB50               DEC  R5
0024 E601               JR   Z,NXT_ONE
0026 E8FB            OD
                  NXT_ONE:

0028 0B04            CP    R4,#%0000       ! CK FOR LAST BLK !
002A 0000
002C E603            JR    Z,WAIT1
002E 0304            SUB   R4,#%0800
0030 0800
0032 E8F0      OD

               WAIT1:
               !*****************************************
                *                                       *
                *  CPU WAITS APPROX. 2MSEC FOR ALL      *
                *  CPU'S TO COMPLETE THE SAME TASK      *
                *  BEFORE CONTINUING.                   *
                *                                       *
                *****************************************!
0034 2103      LD      R3,#50
0036 0032
               DO
0038 1904         MULT  R4,#1
003A 0001
003C AB30         DEC   R3
003E E601         JR    Z,SCRIBE_MEM
0040 F8FB      OD
```

```
                !*****************************************
                *                                       *
                *  SCRIBE_MEM: TRAVERSES THROUGH MEM    *
                *       BLOCKS SEARCHING FOR R/W PAT-   *
                *       TERN INDICATING ACCESSIBLE      *
                *       MEMORY (RAM). CPU INDICATES     *
                *       ITS PRESENCE BY INCREMENTING    *
                *       STORED CPU_NUM. CPU WAITS FOR   *
                *       FOR A PERIOD OF 2MSEC.          *
                *                                       *
                *****************************************!
                SCRIBE_MEM:
0042 2104          LD       R4,#%F800     ! TOP BLOCK ADR !
0044 F800

                ! MAIN LOOP FOR MEMORY SCRIBE !

                   DO
0046 2148             LD     R8,@R4
0048 8B18             CP     R8,R1        ! CK FOR R/W PATTERN !
004A EE09             JR     NZ,NOMATCH   ! NOT FOUND !

                ! CPU INDICATES ACCESS TO THIS BLOCK !
004C A143             LD     R3,R4
004E A931             INC    R3,#2      ! GET CPU_NUM ADR !
0050 3B26             OUT    BUS_LOCK,R2  ! MUTUAL EXCLUSION !
0052 FFF9
0054 2138             LD     R8,@R3       ! GET CPU_NUM !
0056 A980             INC    R8,#1        ! INCREMENT !
0058 2F38             LD     @R3,R8       ! RESTORE !
005A 3B26             OUT    BUS_UNLOCK,R2
005C FFF8

                   NOMATCH:
005E 8D44             TEST   R4
0060 E603             JR     Z,WAIT2    ! FOUND LAST BLOCK !
0062 0304             SUB    R4,#%0800
0064 0800

0066 E8EF          OD
```

```
                WAIT2:
                !*****************************************
                 *                                       *
                 *  CPU WAITS APPROX. 2MSEC FOR ALL      *
                 *  CPU'S TO COMPLETE THE SAME TASK      *
                 *  BEFORE CONTINUING.                   *
                 *                                       *
                 *****************************************!
0068 2103          LD        R3,#50
006A 0032
                   DO
006C 1904             MULT   RR4,#1
006E 0001
0070 AB30             DEC    R3
0072 E601             JR     Z,DEFINE_MEM
0074 F8FB          OD

                !*****************************************
                 *                                       *
                 *  DEFINE_MEM: CPU SEARCHES FOR LOWEST  *
                 *        MEMORY BLOCK WITH HIGHEST CPU  *
                 *        COUNT (GLOBAL) AND LOWEST MEM  *
                 *        BLOCK WITH CPU_NUM=1 (LOCAL).  *
                 *                                       *
                 *  REG USE: RETURN  R7 = LOW GLOBAL     *
                 *                   R6 = LOW LOCAL      *
                 *                   R5 = CPU_NUM (HIGH) *
                 *                                       *
                 *****************************************!
                DEFINE_MEM:
0076 8D58          CLR       R5
0078 2104          LD        R4,#%F800      ! TOP BLOCK ADR !
007A F800
007C A146          LD        R6,R4          ! LOCAL START !
007E A147          LD        R7,R4          ! GLOBAL START !

                   DO
0080 2148             LD     R8,@R4
0082 8B18             CP     R8,R1          ! CK FOR R/W PATTERN !
0084 EE0C             JR     NZ,NEXT_BLK    ! NOT ACCESSIBLE !

                ! CHECK FOR LOCAL OR GLOBAL !
0086 A142             LD     R2,R4
0088 A921             INC    R2,#2          ! GET CPU_NUM ADR !
008A 2123             LD     R3,@R2         ! GET CPU_NUM !
008C 0B03             CP     R3,#1          ! CK FOR LOCAL !
008E 0001
0090 E605             JR     Z,LOCAL_MEM

                ! RECORD GLOBAL MEM AND UPDATE CPU_NUM !
```

```
0092 8B53          CP      R3,R5
0094 E704          JR      C,NEXT_BLK      ! R5 IS HIGHEST !
0096 A135          LD      R5,R3           ! UPDATE CPU CNT !
0098 A147          LD      R7,R4           ! UPDATE GLOBAL ADR !
009A E801          JR      NEXT_BLK

                 ! RECORD LOCAL MEMORY !
                    LOCAL_MEM:
009C A146             LD      R6,R4        ! UPDATE LOCAL !

                    NEXT_BLK:
009E 8D44             TEST    R4
00A0 E603             JR      Z,SPEC_CASE  ! FINISHED !
00A2 0304             SUB     R4,#%0800    ! GET NEXT BLOCK !
00A4 0800

00A6 E8EC          OD

                 !****    R7 = LOW GLOBAL      R6 = LOW LOCAL    ****!
                 !                   R5 = CPU COUNT                  !

                 SPEC_CASE:
                 !****  FOR SINGLE PROCESSOR MONOBOARD CASE  ***!
00A8 2107          LD      R7,#%F800
00AA F800
00AC EE04          JR      NZ,STACK_INT
00AE 2107          LD      R7,#%8000
00B0 8000
00B2 2105          LD      R5,#1
00B4 0001
                 STACK_INT:

                 !*****************************************
                  *                                       *
                  *       LOCAL INITIALIZATION STAGE      *
                  *                                       *
                  *    INITIALIZATION OF INTERNAL  CPU    *
                  *       SPECIAL PURPOSE REGISTERS       *
                  *       AND CPU DATA STRUCTURES.        *
                  *                                       *
                  *****************************************!


                 !*****************************************
                  *  LOAD PROGRAM STATUS AREA POINTER     *
                  *****************************************!

00B6 A161          LD      R1,R6
00B8 0101          ADD     R1,#%0100       ! SET PSA !
00BA 0100
```

```
00BC 7D1D     LDCTL    PSAPOFF,R1     ! LOAD PSAP !

              ! R6 CONTAINS LOW MEMORY AREA FOR DMONITOR DATA!
00BE A164     LD       R4,R6
00C0 0104     ADD      R4,#%0200
00C2 0200
00C4 A14E     LD       R14,R4     ! DATA_AREA !
00C6 34FC     LDAR     R12,ORGADR   ! SET CODE ADR !
00C8 FF36

              ! ****  R12=CODE ADR     R14=DATA AREA  **** !
              !*****************************************
               *           SET STACK POINTER           *
               *****************************************!

00CA A161     LD       R1,R6       ! SET UP STACK PTR !
00CC 0101     ADD      R1,#%00F0
00CE 00F0
00D0 A11F     LD       R15,R1  ! SET SYSTEM STACK POINTER !

              ! ** INITIALIZE REFRESH CNTR REGISTER ** !

00D2 2101     LD       R1,#%9E00    ! LD RATE VALUE !
00D4 9E00
00D6 7D1B     LDCTL    REFRESH,R1

              !*****************************************
               *                                       *
               *  SFTW_INIT: INITIALIZES ALL BASIC     *
               *             DATA STRUCTURES FOR THE   *
               *             SINGLE PROCESSOR.         *
               *                                       *
               *****************************************!

              ! CLEAR DMONITOR RAM AREA !
00D8 A162     LD       R2,R6     ! CLR DMONITOR RAM !
00DA A121     LD       R1,R2
00DC A911     INC      R1,#2
00DE 2103     LD       R3,#%250
00E0 0250
00E2 0D65     LD       @R6,#0
00E4 0000

00E6 BB21     LDIR     @R1,@R2,R3
00E8 0310

              ! INITIALIZE BLANK BUFFERS !
00EA 34E2     LDA      R2,R14(#INTBUF)   ! FILL CONS INPUT BUF !
00EC 0000
00EE 34E1     LDA      R1,R14(#INTBUF) ! WITH SPACES !
```

```
00F0 0000
00F2 A911       INC     R1,#2
00F4 2123       LD      R3,#INSIZ
00F6 0080
00F8 0D25       LD      @R2,#' '
00FA 2020
00FC BB21       LDIR    @R1,@R2,R3
00FE 0310

              ! LD FIXED DATA IN RAM !
0100 34E7       LDA     R7,R14(#INTBUF)
0102 0000
0104 33E7       LD      R14(#INTPTR),R7
0106 0304
0108 34E7       LDA     R7,R14(#OUTBUF)
010A 0080
010C 33E7       LD      R14(#OUTPTR),R7
010E 0306

              !*****************************************
               *   INITIALIZE PROGRAM STATUS AREA      *
               *****************************************!

0110 7D25       LDCTL    R2,PSAPOFF

0112 2101       LD      R1,#%4000
0114 4000
0116 8D48       CLR     R4
0118 3423       LDA     R3,R2(#UNIMP_INST)
011A 0004
                DO
011C 7331          LD   R3(R4),R1
011E 0400
0120 A943          INC  R4,#4
0122 0B04          CP   R4,#28
0124 001C
0126 E601          JR   Z,LD_PC
0128 E8F9       OD
              LD_PC:

012A 3423       LDA     R3,R2(#NMI_INT)    ! LOAD NMI HDLR !
012C 001C
012E 76C4       LDA     R4,NMI(R12)
0130 0000*
0132 3334       LD      R3(#2),R4
0134 0002

                    ! SET INTERRUPT HANDLERS !
0136 3423       LDA       R3,R2(#VEC_PC) ! BASE OF INT VEC!
0138 002A
```

```
013A 76C2      LDA       R2,CONINT(R12) !CONS INPUT!
013C 0000*
013E 3332      LD        R3(#0),R2
0140 0000

                       ! MCZ INPUT !
0142 76C2      LDA       R2,MCZEND(R12)
0144 0000*
0146 3332      LD        R3(#4),R2
0148 0004

            !*****************************************
             *                                       *
             *  HDW_INIT: INITIALIZES HARDWARE IC    *
             *            TO A KNOWN STATE WITH ALL  *
             *            NECESSARY INPUT_OUTPUT     *
             *            FUNCTIONS. INADDITION, THE *
             *            SYSTEM TIMER FUNCTIONS AND *
             *            INTERRUPT STRUCTURE IS SET *
             *            TO A KNOWN STATE.          *
             *                                       *
             *****************************************!

            HDW_INIT:
            ! ** INITIALIZE USARTS  9551 ** !

014A 2101      LD        R1,#%00CE    ! SET MODE !
014C 00CE
014E 3A96      OUTB      PORTBC,RL1
0150 FFE3
0152 3A96      OUTB      PORTAC,RL1
0154 FFDB

0156 2101      LD        R1,#%0027    ! TX, RX, RTS, DTR !
0158 0027
015A 3A96      OUTB      PORTBC,RL1
015C FFE3
015E 3A96      OUTB      PORTAC,RL1
0160 FFDB

            ! ** INIT. INTERRUPT CONTRCLLER  8259A ** !

0162 2101      LD        R1,#%0013    ! LOAD ICW1 !
0164 0013
0166 3A96      OUTB      ICPORT,RL1
0168 FFC9

016A 2101      LD        R1,#%0000    ! LOAD ICW2 !
016C 0000
016E 3A96      OUTB      IDPORT,PL1
```

```
0170 FFCB

0172 2101       LD      R1,#%0003     ! LOAD ICW4 !
0174 0003
0176 3A96       OUTB    IDPORT,PL1
0178 FFCB

              ! ** INITIALIZE TIMING CONTROLLER  9513 ** !

017A 2101       LD      R1,#%FFFF     ! RESET DEVICE !
017C FFFF
017E 3B16       OUT     TCMD, R1
0180 FFD2

0182 2101       LD      R1,#%FFEF     ! SET 16-BIT MODE !
0184 FFEF
0186 3B16       OUT     TCMD,R1
0188 FFD2

018A 2101       LD      R1,#%FF5F     ! LOAD ALL REGS !
018C FF5F
018E 3B16       OUT     TCMD,R1
0190 FFD2

0192 2101       LD      R1,#%FFE8     ! DISABLE AUTO SEQ !
0194 FFE8
0196 3B16       OUT     TCMD,R1
0198 FFD2

              ! SYSTEM 32-BIT CONTINUOUS CLOCK !
019A 2101       LD      R1,#%FF01     ! LD 'CM' CNTR GRP1 !
019C FF01
019E 3B16       OUT     TCMD,R1
01A0 FFD2
01A2 2101       LD      R1,#%0C29     ! CNTR MODE VALUE !
01A4 0C29
01A6 3B16       OUT     TDTA,R1
01A8 FFD0

01AA 2101       LD      R1,#%FF02     ! LD 'CM' CNTR GRP2 !
01AC FF02
01AE 3B16       OUT     TCMD,R1
01B0 FFD2
01B2 2101       LD      R1,#%0029     ! CNTR MODE VALUE !
01B4 0029
01B6 3B16       OUT     TDTA,R1
01B8 FFD0

              ! QUANTUM COUNT-DOWN TIMER !
01BA 2101       LD      R1,#%FF05     ! LD 'CM' CNTR GRP5 !
```

```
01BC FF05
01BE 3B16       OUT         TCMD,R1
01C0 FFD2
01C2 2101       LD          R1,#%0C02     ! CNTR MODE VALUE !
01C4 0C02
01C6 3B16       OUT         TDTA,R1
01C8 FFD0
01CA 2101       LD          R1,#%FF0D     ! LD 'LOAD' GRP5 !
01CC FF0D
01CE 3B16       OUT         TCMD,R1
01D0 FFD2
01D2 2101       LD          R1,#%0005     ! LD VALUE !
01D4 0005
01D6 3B16       OUT         TDTA,R1
01D8 FFD0

              ! HARDWARE PRE-EMPT MECHANISM !
01DA 2101       LD          R1,#%FF04     ! LOAD 'CM' GRP4 !
01DC FF04
01DE 3B16       OUT         TCMD,R1
01E0 FFD2
01E2 2101       LD          R1,#%0C02     ! LOAD CM !
01E4 0C02
01E6 3B16       OUT         TDTA,R1
01E8 FFD0
01EA 2101       LD          R1,#%FF0C     ! LOAD 'LOAD' GRP4 !
01EC FF0C
01EE 3B16       OUT         TCMD,R1
01F0 FFD2
01F2 2101       LD          R1,#%0005     ! LOAD REG !
01F4 0005
01F6 3B16       OUT         TDTA,R1
01F8 FFD0

01FA 2101       LD          R1,#%FF43     ! LD GEN CNTR GRP1&2 !
01FC FF43
01FE 3B16       OUT         TCMD,R1
0200 FFD2
0202 2101       LD          R1,#%FF23     ! ARM REAL TIME CLK !
0204 FF23
0206 3B16       OUT         TCMD,R1
0208 FFD2

020A 7C05       EI          VI
```

```
                !*****************************************
                *                                       *
                *     COORPERATING PROCESSOR STAGE      *
                *              PROPER                   *
                *                                       *
                *  TEST_LOCK: ROUTINE TO GAIN ACCESS    *
                *       TO CONFIG_TABLE.                *
                *                                       *
                *  REG USE: INPUT  R7 = LOW GLOBAL MEM  *
                *                                       *
                *****************************************!
               TEST_LOCK:
020C 7678          LDA      R8,TABLE_LOCK(R7) ! LOCK ADR !
020E 0008

               ! MAIN LOCK TESTING LOOP !
                  DO
0210 3B26            OUT   BUS_LOCK,R2   ! LOCK SYSTEM BUS !
0212 FFF9
0214 0D86            TSET  @R8           ! TEST TABLE LOCK !
0216 3B26            OUT   BUS_UNLOCK,R2 ! UNLOCK SYSTEM BUS !
0218 FFF8
021A ED03            JR    PL,TBL_ACCESS ! GOT EXCLU. ACCESS !

               ! DELAY BEFORE NEXT ATTEMPT TO REDUCE !
               ! CYCLE STEALING ON BUS FROM BOOTLOAD_CPU !
021C 1904            MULT      RR4,#1
021E 0001
0220 E8F7          OD

                !*****************************************
                *                                       *
                *  TBL_ACCESS: ROUTINE TO DETERMINE     *
                *      BOOTLOAD_CPU AND MEMBER_CPU'S.   *
                *                                       *
                *****************************************!
               TBL_ACCESS:
0222 7673          LDA       R3,CPU_CNT(R7)  ! TABLE LOCKED !
0224 000A
0226 2134          LD        R4,@R3          ! GET LOG_CPU NO. !
0228 0B04          CP        R4,#0           ! IS CPU FIRST? !
022A 0000
022C E603          JR        Z,GO_BOOTCPU    ! YES !
022E 76C1          LDA       R1,MEM_CPU(R12)
0230 0000*
0232 1E18          JP        @R1

               GO_BOOTCPU:
0234 76C1          LDA       R1,BOOTLOAD_CPU(R12)
0236 0000*
```

```
0238 1E18      JP       @R1
023A         END INITL
             END BOOTLOAD1
```

2. BOOTLOAD2 MODULE

```
Z8000ASM  2.02
LOC     OBJ CODE     STMT SOURCE STATEMENT

                        1 BOOTLOAD2 MODULE
                $LISTON  $TTY

                CONSTANT

                   RXR          :=  2
                   TXR          :=  0
                   PAR          :=  7
                   PORTAD       :=  %FFD9
                   PORTBD       :=  %FFE1
                   PORTAC       :=  %FFDB
                   PORTBC       :=  %FFE3

                   IDPORT       :=  %FFCB
                   ICPORT       :=  %FFC9

                   ROM_DISABLE  := %FFF0
                   TCMD    :=  %FFD2
                   TDTA    :=  %FFD0

                   BUS_LOCK  :=%FFF9
                   BUS_UNLOCK :=%FFF8
                   VINTR        :=  %(2)0001000000000000
                   VIBIT        :=  12
                   ESCAPE       :=  %1B
                   BS           :=  %08
                   LINDEL       :=  %7F
                   CR           :=  %0D
                   LF           :=  %0A
                   TXOFCH       :=  %13
                   TXONCH       :=  %11
                   INSIZ        :=  128        ! INTBUF SIZE !
                   OUTSIZ       :=  128        ! OUTBUF SIZE !
                   RBSIZ        :=  256        ! RING BUFFER SIZE !
                ! BIT POSITIONS IN MONITOR FLAG WORD !
                   TRPMDE       :=  0
                   ISTOP        :=  1
                   OSTOP        :=  2
                   SNDMDE       :=  3
                   LDMDE        :=  4
                   ESC          :=  5
                   TXMSK        := %6

                   MAX_CPU      :=  8
```

```
                TYPE
                   MESSAGE    ARRAY[3 WORD]
                   MEM_ARRAY  ARRAY[32 WORD]
                   SWITCH     ARRAY[2 WORD]
                   CPU_ENTRY        RECORD [
                            SIGNAL          WORD
                            CPU_ID          WORD
                            MSG_BLK         MESSAGE
                            MEM_MAP         MEM_ARRAY]
                   ID_ARRAY          ARRAY[MAX_CPU WORD]
                   ENTRY_ARRAY       ARRAY[MAX_CPU CPU_ENTRY]

                INTERNAL
                $SECTION TABLE1_DATA
                $ABS 0

0000               CONFIG_TABLE    RECORD [
                            RW_PATTERN      WORD
                            CPU_NUM         WORD
                            NORM_RW_PAT     WORD
                            NORM_CPU_CNT    WORD
                            TABLE_LOCK      WORD
                            CPU_CNT         WORD
                            CPU_LIST        ENTRY_ARRAY]



                INTERNAL
                $SECTION DATA_DEC
                $ABS 0
0000               INTBUF    ARRAY [128 BYTE]
0080               OUTBUF    ARRAY [128 BYTE]
0100               RNGBUF    ARRAY [256 BYTE]
0200               MCZBUF    ARRAY [256 BYTE]

0300               BUFADR    WORD
0302               BUFSIZ    WORD
0304               INTPTR    WORD
0306               OUTPTR    WORD
0308               NXTPTR    WORD
030A               GETOUT    WORD
030C               MCZPUT    WORD
030E               MCZGET    WORD

0310               MFLAGS    WORD

0312               RETRY     WORD
0314               ADR_STR   WORD
```

```
                INTERNAL
                $SECTION PSA_DATA
                $ABS 0
0000               PSA    RECORD [
                             DATA_AREA      WORD
                             CODE_AREA        WORD
                             UNIMP_INST     SWITCH
                             PRIV_INST      SWITCH
                             SYSTEM_CALL    SWITCH
                             SEG_TRAP       SWITCH
                             NMI_INT        SWITCH
                             NVI_INT        SWITCH
                             VEC_FCW        WORD
                             VEC_PC         ARRAY [200  WORD]
                             ]

                $SECTION BOOT2_PROC
                EXTERNAL
                   NMI        PROCEDURE
                   GETLNE     PROCEDURE
                   SNDCHR     PROCEDURE
                   SNDMSG     PROCEDURE
                   MCZHND     PROCEDURE
                   CONINT     PROCEDURE
                   GETBUF     PROCEDURE
                   QUIT       PROCEDURE
                   LOAD_FILE PROCEDURE
                   PHYS_ID      LABEL
```

```
                  ! *****     BOOTLOAD_CPU  START     *****  !
                  GLOBAL
                  NMI_RTN      LABEL
                  FROP         LABEL
                  MEM_CPU      LABEL
0000              BOOTLOAD_CPU  PROCEDURE
                  !*****************************************
                   *                                       *
                   *  BOOTLOAD_CPU: CPU ASSUMES ROLE AS    *
                   *       INITIALIZATION COORDINATOR.     *
                   *       CPU DETERMINES NUMBER OF CPU    *
                   *       IN SYSTEM AND INITIALIZES THE   *
                   *       CONFIG_TABLE ACCORDINGLY. IT    *
                   *       THEN IDENTIFIES ITSELF AND      *
                   *       WAITS FOR MEMBER_CPU'S TO       *
                   *       IDENTIFY THEMSELVES; AND THEN   *
                   *       CONSOLIDATES CONFIG_TABLE DATA  *
                   *       INTO SYSTEM TABLE (MIN_CONFIG_  *
                   *       TBL).                           *
                   *                                       *
                   *  REG USE: INPUT  R5 = NUMBER CPU'S    *
                   *                  R6 = LOW LOCAL MEM   *
                   *                  R7 = LOW GLOBAL MEM  *
                   *                                       *
                   *****************************************!
                  ENTRY
                  ! LOAD CPU_CNT WITH NEXT LOG_CPU NUMBER !
0000 4D75            LD        CPU_CNT(R7),#1    ! LOAD LOG_CPU 1 !
0002 000A
0004 0001
0006 4DE5            LD        ADR_STR(R14),#0   ! SAVE LOG_CPU NO. !
0008 0314
000A 0000

                  ! CLEAR CONFIG_TABLE FOR CPU_NUM ENTRIES !
000C A153            LD        R3,R5     ! GET CPU COUNT !
000E 8D28            CLR       R2
                     DO
0010 0102               ADD    R2,#37    ! SIZE OF CPU ENTRY !
0012 0025
0014 AB30               DEC    R3
0016 E601               JR     Z,NXT
0018 E9FB            OD

                  ! CLEAR CONFIG_TABLE FOR TOTAL CPU ENTRIES !
                  NXT:
001A 7673            LDA       R3,CPU_LIST(R7)  ! BASE ADR !
001C 000C
001E 0D35            LD        @R3,#0
0020 0000
```

```
0022 A134       LD      R4,R3
0024 0941       INC     R4,#2
0026 BB31       LDIR    @R4,@R3,R2   ! CLEAR TABLE !
0028 0240

                ! COMPLETE OWN CONFIG_TABLE ENTRY !
002A 7673       LDA     R3,CPU_LIST(R7)    ! LOG_CPU 0 !
002C 000C
002E 61C8       LD      R8,PHYS_ID(R12)    ! UNIQUE ID !
0030 0000*
0032 6F38       LD      CPU_ID(R3),R8 ! ENTER IN TABLE !
0034 0002
0036 763D       LDA     R13,MEM_MAP(R3)
0038 000A
003A DF1C       CALR    MAP_MEMORY   ! ENTER MEM MAP !

                ! UNLOCK CONFIG_TABLE !
003C 7678       LDA     R8,TABLE_LOCK(R7) ! LOCK ADR !
003E 0008
0040 0D85       LD      @R8,#0         ! CLEAR LOCK !
0042 0000

                ! WAIT FOR ALL CPU'S TO IDENTIFY THEMSELVES !
0044 7678       LDA     R8,CPU_CNT(R7)  ! CPU_CNT ADR !
0046 000A
0048 A159       LD      R9,R5       ! GET TOTAL PHYS_CPU !
                DO
004A 218D           LD      R13,@R8     ! GET VALUE !
004C 8B9D           CP      R13,R9      ! CK FOR MATCH !
004E E603           JR      Z,GOT_SIG
0050 1904           MULT    RR4,#1      ! DELAY !
0052 0001
0054 E8FA       OD

                GOT_SIG:
0056 DF53       CALR    SIGNAL_CPU
0058 E82A       JR      DECISION    ! PROCEDE TO SECONDARY !
                                    ! STORAGE INTERFACING  !
```

```
                ! *****      MEMBER_CPU  START    ***** !
                !*********************************************
                 *                                           *
                 *      MEM_CPU: IDENTIFIES ITSELF IN        *
                 *        CONFIG_TABLE AND ENTERS MEM_       *
                 *        MAP. WAITS FOR DIRECTOR_CPU        *
                 *        TO SIGNAL TO CONTINUE.             *
                 *                                           *
                 *  REG USE: INPUT  R3 = BASE OF OWN         *
                 *                       ENTRY               *
                 *                                           *
                 *********************************************!
                MEM_CPU:
005A 7673          LDA       R3,CPU_LIST(R7)  ! BASE OF ENTRY !
005C 000C
005E 6174          LD        R4,CPU_CNT(R7)   ! LOG_CPU NUMBER !
0060 0008
0062 6FE4          LD        ADR_STR(R14),R4  ! SAVE LOG_CPU NO. !
0064 0314
0066 8D28          CLR       R2
                   DO
0068 0102             ADD    R2,#74          ! DETERMINE BASE OF !
006A 004A
006C AB40             DEC    R4              ! ENTRY !
006E E601             JR     Z,CONT
0070 E8FB          OD

                CONT:
0072 812A          ADD       R10,R2        ! COMPUTE BASE ADR !
0074 A1A3          LD        R3,R10     ! SAVE BASE ADR !

                ! ENTER UNIQUE ID IN TABLE ENTRY !
0076 76C2          LDA       R2,PHYS_ID(R12)
0078 0000*
007A 2128          LD        R8,@R2          ! GET UNIQUE ID !
007C 6F38          LD        CPU_ID(R3),R8 ! ENTER IN TABLE !
007E 0002

                ! ENTER MEM_MAP IN TABLE ENTRY !
0080 763D          LDA       R13,MEM_MAP(R3)  ! MEM_MAP BASE !
0082 000A
0084 DF41          CALR      MAP_MEMORY

                ! LOAD CPU_CNT WITH NEXT LOG_CPU NUMBER !
0086 7678          LDA       R8,CPU_CNT(R7)
0088 000A
008A 2189          LD        R9,@R8          ! CPU COUNT !
008C A990          INC       R9              ! ADD ONE, !
008E 2F89          LD        @R8,R9          ! AND PUT BACK !
```

```
                ! UNLOCK CONF_TABLE FOR OTHER PROCESSORS !
0090 7678       LDA     R8,TABLE_LOCK(R7)
0092 0008
0094 0D85       LD      @R8,#0          ! CLEAR LOCK !
0096 0000

                ! WAIT FOR SIGNAL TO PROCEED WITH BOOTLOAD !
0098 7638       LDA     R8,SIGNAL(R3)   ! OWN SIGNAL ADR !
009A 0000
                DO
009C 2182           LD      R2,@R8      ! TEST SIGNAL !
009E 0302           CP      R2,#1
00A0 0001
00A2 E603           JR      Z,RESET_SIG
00A4 1904           MULT    RR4,#1      ! DELAY !
00A6 0001
00A8 F8F9       OD

                RESET_SIG:
00AA 4D38       CLR     SIGNAL(R3)
00AC 0000

                !*****************************************
                 *                                       *
                 *       BOOTSTRAP LOADING FUCTION       *
                 *                                       *
                 *   THIS SECTION CONTAINS NECESSARY     *
                 *   SECONDARY STORAGE  INTERFACING      *
                 *   PRIMITIVES TO EFFECT A DOWNLOAD.    *
                 *                                       *
                 *****************************************!


                !*****************************************
                 *   BOOTLOAD-MONITOR DECISION POINT     *
                 *****************************************!

                DECISION:
00AE C82A       LDB     RL0,#'*'        ! PROMPT MIN MBC !
                                        ! INITIALIZATION !
00B0 76C4       LDA     R10,SNDCHR(R12)
00B2 0000*
00B4 1FA0       CALL    @R10            ! OUTPUT PROMPT !

                ! SET RINGBUFFER OFFSET POINTERS !
00B6 4DF8       CLR     GETOUT(R14)
00B8 030A
00BA 4DE8       CLR     NXTPTR(R14)
00BC 0308
```

```
00BE 7C05       EI      VI
                ! GLOBAL TABLE FOR CPU SYNCHRONIZATION; LOG_CPU !
                ! NUMBER DETERMINES BOOT_COORDINATOR IN GENERAL    !
                ! CASE, BUT NOT USED WITH MCZ IMPLEMENTATION.    !

                ! MAIN LOOP FOR BOOT_COORD DETERMINATION !
                   DO
                   ! CHECK FOR CONSOLE INPUT !
00C0 61E0          LD      R0,GETOUT(R14)
00C2 030A
00C4 4BE0          CP      R0,NXTPTR(R14)
00C6 0308
00C8 EE06          JR      NZ,RESPONSE    ! HAVE REC CHR !

                   ! CHECK FOR BOOTLOAD SIGNAL !
                LOP_MOR:
00CA 6131          LD      R1,SIGNAL(R3)
00CC 0000
00CE 0B01          CP      R1,#%0001    ! CK FOR SIGNAL !
00D0 0001
00D2 E62C          JR      Z,BGN_BOOT    ! REC BOOTLOAD SIG !
00D4 E8F5       CD

                RESPONSE:
                ! DETERMINE INPUT CHAR RESPONSE !
00D6 76CA       LDA     R10,GETBUF(R12)
00D8 0000*
00DA 1FA0       CALL    @R10
00DC 2028       LDB     RL0,@R2
00DF 76CA       LDA     R10,SNDCHR(R12)
00E0 0000*
00E2 1FA0       CALL    @R10        ! ECHO TO CONSOLE !
00E4 A10D       LD      R13,R0      ! SAVE CHAR !
00E6 C80D       LDB     RL0,#CR
00F8 1FA0       CALL    @R10        ! SEND CR TO CONS !
00EA C80A       LDB     RL0,#LF
00EC 1FA0       CALL    @R10        ! SEND LINE FEED !

                ! INSURE MCZ SYSTEM IS INITIALIZED !
00EE 3402       LDAR    R2,MCZ_MSG     ! INIT MESSAGE !
00F0 00F0
00F2 76CA       LDA     R10,SNDMSG(R12)
00F4 0000*
00F6 1FA0       CALL    @R10         ! SEND TO CONSOLE !

00F8 4DF8       CLR     GETOUT(R14)
00FA 030A
00FC 4DE8       CLR     NXTPTR(R14)
00FE 0308
0100 76CA       LDA     R10,QUIT(R12)
```

```
0102 0000*
0104 1E08     JP        @R10          ! ENTER TRANSPARENT MODE !

              ! RETURN POINT FORM TRANSPARENT MODE !
              NMI_RTN:
              ! DETERMINE COMMAND BRANCH !
0106 3402     LDAR      R2,LOAD_MSG
0108 0086
010A 76CA     LDA       R10,SNDMSG(R12)
010C 0000*
010F 1FA0     CALL      @R10

0110 A1D2     LD        R2,R13
0112 0A0A     CPB       RL2,#%53      ! CK FOR SASS BOOTSTRAP !
0114 5353
0116 EE03     JR        NZ,OTHER
0118 340D     LDAR      R13,BOOT_COORD
011A 021E
011C E802     JR        SET_PTRS

              OTHER:
011E 340D     LDAR      R13,MONITOR   ! START OF DMONITOR !
0120 007C

              SET_PTRS:
0122 61E1     LD        R1,NXTPTR(R14)  ! RESET RNIGBUF !
0124 0308
0126 6FE1     LD        GETOUT(R14),R1  ! OFFSET POINTERS !
0128 030A

012A 1ED8     JP        @R13          ! BRANCH !


              ! NON-BOOTLOAD_CPU RESPONSE TO SIGNAL !
              BGN_BOOT:
              ! PASS LOG_CPU OF CPU INTO BOOTSTRAP !
012C 61FC     LD        R12,ADR_STR(R14)
012E 0314
0130 4D38     CLR       SIGNAL(R3)
0132 0000

0134 6131     LD        R1,MSG_BLK(R3)
0136 0004
0138 1E18     JP        @R1       ! TRANSFER TO BOOTSTRAP !
```

```
                !****************************************
                *   ASSUME BOOTLOAD COORDINATOR ROLE     *
                ****************************************!
                BOOT_COORD:
013A 6FE3         LD        RETRY(R14),R3  ! SAVE ENTRY BASE !
013C 0312
013E 3404         LDAR      R4,BOOTLOAD  ! LOAD FILENAME !
0140 0090
0142 A173         LD        R3,R7       ! GLOBAL MEMORY !
0144 0103         ADD       R3,#%0200  ! BOOTLOAD ADR !
0146 0200

                ! LOAD BOOTLOAD PROGRAM !
0148 76CA         LDA       R10,LOAD_FILE(R12)
014A 0000*
014C 1FA0         CALL      @R10

014E A174         LD        R4,R7       ! GLOBAL MEMORY !
0150 414B         LD        R11,R4      ! SAVE BASE ADR !
0152 010B         ADD       R11,#%0200  ! BOOTLOAD ADR !
0154 0200
0156 210C         LD        R12,#0
0158 0000
015A A151         LD        R1,R5             ! NUMBER OF CPU'S !

                ! STORE BOOTLOAD ADDRESS IN MSG_BLK OF ALL CPU'S !
                  DO
015C 6F43           LD      MSG_BLK(R4),R3   ! PLACE ADDRESS !
015E 0004
0160 AB10           DEC     R1
0162 E603           JR      Z,SIG_NXT
0164 0104           ADD     R4,#74           ! NEXT LOG_CPU NUMBER !
0166 004A
0168 E8F9         OD

                SIG_NXT:
                ! SIGNAL ALL CPU'S TO TRANSFER CONTROL !
016A 61EC         LD        R12,ADR_STR(R14)   ! GET OWN LOG_CPU NO. !
016C 0314
016E DFDF         CALR      SIGNAL_CPU
0170 617A         LD        R10,CPU_LIST(R7)
0172 000C
0174 4DA5         LD        SIGNAL(R10),#1     ! BOOTLOAD CPU SIGNAL !
0176 0000
0178 0001
017A 61E3         LD        R3,RETRY(R14)      ! OWN ENTRY BASE !
017C 0312
017E 4D38         CLR       SIGNAL(R3)
0180 0000
0182 1FB9         JP        @R11            ! TRANSFER SELF !
```

```
                ERROR:
0184 3402           LDAR    R2,ERR_MSG
0186 0074
0188 76CA           LDA     R10,SNDMSG(R12)
018A 0000*
018C 1FA0           CALL    @R10
018E E86F           JR      DECISION


                !********** END BOOTLOAD ****************!

                LOAD_MSG:
0190 0D             BVAL    %0D
0191 4C             BVAL    'L'
0192 4F41           WVAL    'OA'
0194 4449           WVAL    'DI'
0196 4E47           WVAL    'NG'
0198 2E2E           WVAL    '..'
019A 2E2E           WVAL    '..'
019C 2E             BVAL    '.'
019D 0D             BVAL    %0D

                MONITOR:
019F 3404           LDAR    R4,DMONITOR   ! LOAD FILE NAME !
01A0 004C
01A2 2103           LD      R3,#%0000
01A4 0000
01A6 76CA           LDA     R10,LOAD_FILE(R12)
01A8 0000*
01AA 1FA0           CALL    @R10
01AC 2103           LD      R3,#%0000
01AE 0000
01B0 F84C           JR      DISABLE_PROM  ! TRANSFER CONTROL !

01B2            END  BOOTLOAD_CPU
```

```
01B2          SIGNAL_CPU  PROCEDURE
              !*****************************************
               *                                       *
               *  SIGNAL_CPU: PLACES SIGNAL (#1) IN    *
               *       SIGNAL BLOCK FOR EACH CPU EN-   *
               *       TRY IN CONFIG_TABLE.            *
               *                                       *
               *****************************************!
              ENTRY
              ! SIGNAL ALL CPU TO DOWN-LOAD !
01B2 767D        LDA       R13,CPU_LIST(R7)
01B4 000C
01B6 010D        ADD       R13,#74        ! LOG_CPU 1 ENTRY !
01B8 004A
01BA A154        LD        R4,R5          ! TOTAL NO. CPU !
01BC AB40        DEC       R4
01BE E608        JR        Z,ALL_SIG

                 DO
01C0 4DD5           LD     SIGNAL(R13),#1  ! LOAD SIGNAL !
01C2 0000
01C4 0001
01C6 AB40           DEC    R4
01C8 E603           JR     Z,ALL_SIG
01CA 010D           ADD    R13,#74        ! NEXT LOG_CPU ADR !
01CC 004A
01CE E8F8        OD

              ALL_SIG:
01D0 9E08        RET

01D2          END SIGNAL_CPU

              BOOTLOAD:
01D2 0F          BVAL      %0F
01D3 4C          BVAL      'L'
01D4 4F41        WVAL      'OA'
01D6 4420        WVAL      'D '
01D8 424F        WVAL      'BO'
01DA 4F54        WVAL      'OT'
01DC 5354        WVAL      'ST'
01DE 5241        WVAL      'RA'
01E0 5020        WVAL      'P '

              MCZ_MSG:
01E2 0A          BVAL      %0A
01E3 52          BVAL      'R'
01E4 4553        WVAL      'ES'
01E6 4554        WVAL      'ET'
01E8 204D        WVAL      ' M'
```

```
01F8 435A       WVAL    'CZ'
01EC 0D         BVAL    %0D
01ED 20         BVAL

                DMONITOR:
01EE 0D         BVAL    %0D
01EF 4C         BVAL    'L'
01F0 4F41       WVAL    'OA'
01F2 4420       WVAL    'D '
01F4 4D4F       WVAL    'MO'
01F6 4E49       WVAL    'NI'
01F8 544F       WVAL    'TO'
01FA 5220       WVAL    'R '

                ERR_MSG:
01FC 06         BVAL    %06
01FD 45         BVAL    'E'
01FE 5252       WVAL    'RR'
0200 4F52       WVAL    'OR'
0202 0D         BVAL    %0D
0203 20         BVAL

0204            MAP_MEMORY  PROCEDURE
                !*****************************************
                *                                       *
                *  MAP_MEMORY: MAPS CPU MEMORY ACCESS   *
                *        BY DOMAIN, AS TO LOCAL (1),    *
                *        GLOBAL (2), DUAL_USE (3),      *
                *        NON_USE (4), NON_ACCESS (5).   *
                *                                       *
                *  REG USE: INPUT  R13 = ADR MEM_MAP    *
                *                                       *
                *****************************************!
                ENTRY
0204 A1D1       LD      R1,R13          ! SAVE BASE ADR !
0206 2102       LD      R2,#%55AA       ! R/W PATTERN !
0208 55AA
020A 8D48       CLR     R4
020C 2109       LD      R9,#%F800       ! END ADR !
020E F800
0210 2110       LD      R0,@R1          ! 1ST MAP BLK !
0212 DFFF       CALR    SYS_MAP     ! MAP SYSTEM MODE !
0214 9E08       RET
0216            END MAP_MEMORY
```

```
0216            SYS_MAP  PROCEDURE
                !*****************************************
                *                                       *
                *  SYS_MAP: MAPS MEMORY ACCESS IN THE   *
                *     SYSTEM MODE INTO CONFIG_TABLE     *
                *     FOR CPU.                          *
                *                                       *
                *  REG USE: INPUT  R1 = BASE MAP BLK    *
                *                  R2 = R/W PATTERN     *
                *                  R4 = START MEM       *
                *                  R9 = END ADR         *
                *                                       *
                *****************************************!
                ENTRY
                  DO
0216 6148           LD      R8,RW_PATTERN(R4)      ! CK FOR R/W !
0218 0000
021A 8B28           CP      R8,R2
021C EE0D           JR      NZ,BAD_MEM  ! NO R/W/ !
021E 6148           LD      R8,CPU_NUM(R4)     ! GET CPU_CNT !
0220 0002
0222 0B08           CP      R8,#%0001  ! CK FOR LOCAL !
0224 0001
0226 EE02           JR      NZ,GLOBE_CK
0228 C001           LDB     RH0,#%01   ! RECORD LOCAL !
022A E807           JR      CONTINUE

                GLOBE_CK:
022C 8B58           CP      R8,R5      ! CK FOR GLOBAL !
022E EE02           JR      NZ,NO_USE  ! 1<CPU_NUM<R5 !
0230 C002           LDB     RH0,#%02    ! RECORD GLOBAL !
0232 E803           JR      CONTINUE

                NO_USE:
0234 C004           LDB     RH0,#%04    ! RECORD NON_USE !
0236 E801           JR      CONTINUE

                BAD_MEM:
0238 C005           LDB     RH0,#%05    ! RECORD NON_ACCESS !

                CONTINUE:
023A 2F10           LD      @R1,R0      ! STORE MAP BLOCK !
023C A911           INC     R1,#2       ! NEXT MAP BLK !
023F 8B49           CP      R9,R4
0240 9E06           RET     Z           ! FINISHED !
0242 9D08           CLR     R0
0244 0104           ADD     R4,#%0800   ! NEXT MEM ADR !
0246 0800

0248 E8F6       OD
```

```
024A                END SYS_MAP


024A                DISABLE_PROM  PROCEDURE
                    !*****************************************
                     *                                       *
                     *  DISABLE_PROM: REPOSITIONS CODE FOR   *
                     *          PROM DISABLING AND TRANSFER  *
                     *          OF CONTROL FLOW.             *
                     *                                       *
                     *  REG USE: INPUT  R3 = TRANS ADR       *
                     *                                       *
                     *****************************************!
                    ENTRY
024A A161              LD         R1,R6       ! MOVE LOCATION !
024C 3404              LDAR       R4,BGN_CODE
024E 0014
0250 2102              LD         R2,#3
0252 0003
                    ! ACTUAL CODE TRANSFER !
                       DO
0254 214D                LD       R13,@R4     ! GET INSTRUCTION !
0256 2F1D                LD       @R1,R13     ! PUT AT NEW LOC !
0258 AB20                DEC      R2
025A E603                JR       Z,DO_CODE
025C A911                INC      R1,#2
025E A941                INC      R4,#2
0260 E8F9              OD

                    DO_CODE:
0262 1E68              JP         @R6         ! TRANSFER CONTROL !

                    BGN_CODE:
0264 3B16              OUT        ROM_DISABLE,R1
0266 FFF0
0268 1E38              JP         @R3

026A                END  DISABLE_PROM

                    END BOOTLOAD2
```

3. SUPPORT1 MODULE

```
Z8000ASM  2.02
LOC     OBJ CODE     STMT SOURCE STATEMENT

                       1 SUPPORT1 MODULE
                $LISTON  $TTY

                !*****************************************
                 *                                       *
                 *  SUPPORT1 MODULE: MODULE ONE FOR      *
                 *        SECONDARY STORAGE PRIMITIVE    *
                 *        FUNCTIONS SUPPORT. STRICITLY   *
                 *        HARDWARE DEPENDENT; SHOULD     *
                 *        MEET STORAGE DEVICE REQUIRE-   *
                 *        MENTS FOR INTERFACING.         *
                 *                                       *
                 *****************************************!

                CONSTANT

                   RXR        :=  2
                   TXR        :=  0
                   PAR        :=  7
                   PORTAD     :=  %FFD9
                   PORTBD     :=  %FFE1
                   PORTAC     :=  %FFDB
                   PORTBC     :=  %FFE3

                   IDPORT     :=  %FFCB
                   ICPORT     :=  %FFC9

                   TCMD    :=  %FFD2
                   TDTA    :=  %FFD0

                   BUS_LOCK :=%FFF9
                   BUS_UNLOCK :=%FFF8
                   VINTR      :=  %(2)0001000000000000
                   VIBIT      :=  12
                   ESCAPE     :=  %1B
                   BS         :=  %08
                   LINDEL     :=  %7F
                   CR         :=  %0D
                   LF         :=  %0A
                   TXOFCH     :=  %13
                   TXONCH     :=  %11
                   INSIZ      :=  128        ! INTBUF SIZE !
                   OUTSIZ     :=  128        ! OUTBUF SIZE !
                   RBSIZ      :=  256        ! RING BUFFER SIZE !
```

```
                    ! BIT POSITIONS IN MONITOR FLAG WORD !
                       TRPMDE    :=  0
                       ISTOP     :=  1
                       OSTOP     :=  2
                       SNDMDE    :=  3
                       LDMDE     :=  4
                       ESC       :=  5
                       TXMSK     := %6

                       COMDS     :=  12

                    EXTERNAL   GETNXT    PROCEDURE
                    EXTERNAL   GETADR    PROCEDURE
                    EXTERNAL   GMCZAD    PROCEDURE
                    EXTERNAL   SNDMCZ    PROCEDURE
                    EXTERNAL   GETBUF    PROCEDURE
                    EXTERNAL   SNDCHR    PROCEDURE
                    EXTERNAL   EROR      LABEL
                    EXTERNAL   SKPBLK    PROCEDURE
                    EXTERNAL   CONVERT   PROCEDURE
                    EXTERNAL   GETCHR    PROCEDURE

                    INTERNAL
                    $SECTION DATA_DEC
                    $ABS 0

0000                   INTBUF    ARRAY [128 BYTE]
0080                   OUTBUF    ARRAY [128 BYTE]
0100                   RNGBUF    ARRAY [256 BYTE]
0200                   MCZBUF    ARRAY [256 BYTE]

0300                   BUFADR    WORD
0302                   BUFSIZ    WORD

0304                   INTPTR    WORD
0306                   OUTPTR    WORD

0308                   UNIMP     WORD
030A                   BRKCNT    WORD

030C                   NXTPTR    WORD
030E                   GETOUT    WORD
0310                   MCZPUT    WORD
0312                   MCZGET    WORD
0314                   BRKSTR    WORD
0316                   BRKADR    WORD
0318                   TMPSP     WORD
031A                   TMPFCW    WORD

031C                   MFLAGS    WORD
```

```
                ! USER REGISTER STORAGE !

031F            R0_       WORD
0320            R1_       WORD
0322            R2_       WORD
0324            R3_       WORD
0326            R4_       WORD
0328            R5_       WORD
032A            R6_       WORD
032C            R7_       WORD
032E            R8_       WORD
0330            R9_       WORD
0332            R10_      WORD
0334            R11_      WORD
0336            R12_      WORD
0338            R13_      WORD
033A            R14_      WORD
033C            R15_      WORD
033E            RPC_      WORD
0340            RFC_      WORD

0342            RETRY     WORD

                GLOBAL
                $SECTION SUPPORT1_PROC
                $REL 0

                GLOBAL
0000            GETLNE  PROCEDURE
                !*****************************************
                 *                                       *
                 *  GETLNE: REC ONE LINE INPUT FROM      *
                 *          CONS (PORT2), UP TO 80-CHR   *
                 *          MAX, STORE IN INTBUF PLUS    *
                 *          CR, AND ECHO BACK TO CONS.   *
                 *                                       *
                 *  REG USE: RETURN  RL0= 1ST CHR IN BUF*
                 *                   AND Z IF CHR = CR   *
                 *                                       *
                 *****************************************!
                ENTRY
0000 76F2         LDA       R2,INTBUF(R14)   ! GET BASE INTBUF !
0002 0000
0004 2101         LD        R1,#INSIZ    ! GET MAX SIZE !
0006 0080
0008 6FF2         LD        INTPTR(R14),R2
000A 0304
000C DFF8         CALR      CONSOL       ! FILL LINE IN INTBUF!
000E 34CA         LDA       R10,R12(#SKPBLK)
```

```
0010 011C'
0012 1FA0       CALL    @R10
0014 6BE0       DEC     INTPTR(R14)       ! RETURN TO START !
0016 0304
0018 0AC8       CPB     RL0,#CR
001A 0D0D
001C 9E08       RET                       ! GOT 1ST CHR = CR !
001E            END GETLNE

                GLOBAL
001E            CONSOL  PROCEDURE
                !*****************************************
                 *                                       *
                 *  CONSOL: STORE CONS INPUT LINE IN     *
                 *          BUFFER ADDRESS PROVIDED,     *
                 *          PLACE CR AT END OF LINE,     *
                 *          AND PROVIDE DELETE CHAR      *
                 *          AND DELETE LINE EDIT FUNC.   *
                 *                                       *
                 *  REG USE: INPUT  R1 = SIZE OF BUFFER  *
                 *                  R2 = BUFFER ADR      *
                 *           RETURN R1 = # OF REC CHR    *
                 *                  AND Z IF BUF LIMIT   *
                 *                                       *
                 *****************************************!
                ENTRY
001E 6FE2       LD      BUFADR(R14),R2    ! SAVE BOTH BUF ADR !
0020 0300
0022 6FE1       LD      BUFSIZ(R14),R1    ! AND BUF SIZE !
0024 0302
                HDN0:
0026 61E2       LD      R2,BUFADR(R14)
0028 0300
002A 8D18       CLR     R1
                REDLOP:
002C DFC1       CALR    CONRD     ! GET CHAR AND ECHO !
002E 0A08       CPB     RL0,#%61  ! CONVERT TO UPPER CASE!
0030 6161
0032 E704       JR      C,UPCASE  ! NOT LOWER CASE !
0034 0A08       CPB     RL0,#%7B
0036 7B7B
0038 EF01       JR      NC,UPCASE ! YES LOWER CASE !
003A A285       RESB    RL0,#5    ! CONVERT TO UPCASE !
                UPCASE:
003C 2E28       LDB     @R2,RL0

                ! PERFORM EDIT FUNCTIONS ON INPUT !
003E 0A08       CPB     RL0,#BS   !CK FOR DEL CHR !
0040 0809
0042 EE11       JR      NZ,CONTCK    ! NO, CONTINUE CK !
```

```
0044 A820       DEC     R2,#1           ! YES, BACKSPACE !
0046 AB10       DEC     R1
0048 4BF2       CP      R2,BUFADR(R14)      ! NOT TOO FAR !
004A 0300
004C E707       JR      C,DO_OVR
004E C820       LDB     RL0,#' '
0050 34CA       LDA     R10,R12(#SNDCHR)
0052 01DE'
0054 1FA0       CALL    @R10            ! BLANK OUT BAD CHR !
0056 C808       LDB     RL0,#BS
0058 1FA0       CALL    @R10
005A E8E8       JR      REDLOP      ! CONTINUE.....!

            DO_OVR:
005C C82A       LDB     RL0,#'*'
005E 34CA       LDA     R10,R12(SNDCHR)
0060 01DE
0062 1FA0       CALL    @R10            ! SEND PROMPT !
0064 E8F2       JR      HDNG            ! START AGAIN !

            CONTCK:
0066 0A08       CPB     RL0,#LINDEL     ! CK FOR LINE DEL !
0068 7F7F
006A E609       JR      Z,DELINE        ! YES,.....!
006C A920       INC     R2,#1
006E A910       INC     R1
0070 0A08       CPB     RL0,#CR         ! CK FOR CR !
0072 0D0D
0074 E60E       JR      Z,ADDLF         ! YES, ADD LF CHR !
0076 4BE1       CP      R1,BUFSIZ(R14)      ! SIZE CK !
0078 0302
007A EED8       JR      NZ,REDLOP       ! OK, GET NEXT CHR !
007C 9E06       RET     Z               ! TOO LARGE, ERROR !

            DELINE:
007E C85E       LDB     RL0,#%5E
0080 34CA       LDA     R10,R12(SNDCHR) ! SND LINE DEL !
0082 01DE'
0084 1FA0       CALL    @R10
0086 DFF5       CALR    NEWLNE          ! START NEW LINE !
0088 C820       LDB     RL0,#' '
008A 34CA       LDA     R10,R12(SNDCHR)
008C 01DE
008E 1FA0       CALL    @R10            ! SND CHR !
0090 E8CA       JR      HDNG          ! START AGAIN !

            ADDLF:
0092 C80A       LDB     RL0,#LF
0094 34CA       LDA     R10,R12(#SNDCHR)
0096 01DE
```

```
0098 1FA0      CALL     @R10          ! SEND LF CHR !
009A 8D43      RESFLG   Z
009C 9E08      RET
009E         END CONSOL

             GLOBAL
009E         NEWLNE  PROCEDURE
             !*****************************************
              *                                       *
              *  NEWLNE: SENDS CR AND LF TO CONSOLE   *
              *                                       *
              *****************************************!
             ENTRY
009E 34CA      LDA      R10,R12(SNDCHR)
00A0 01DE
00A2 C80D      LDB      RL0,#CR
00A4 1FA0      CALL     @R10
00A6 C80A      LDB      RL0,#LF
00A8 1FA0      CALL     @R10      ! ADR SNDCHR IN R10 !
00AA 9E08      RET
00AC         END NEWLNE

             GLOBAL
00AC         CONRD  PROCEDURE
             !*****************************************
              *                                       *
              *  CONRD: GETS CHAR FROM CONSOLE INPUT  *
              *         BUFFER (INTBUF) AND ECHOS     *
              *         BACK TO CONSOLE. LOOPS UNTIL  *
              *         RECEIVE CHARACTER.            *
              *                                       *
              *  REG USE: RETURN  RL0= CHR            *
              *                   AND Z IF CHR=CR     *
              *                                       *
              *****************************************!
             ENTRY
00AC 61E0    TC:LD      R0,GETOUT(R14)
00AE 030E
00B0 4BE0      CP       R0,NXTPTR(R14)    ! COMPARE GET AND !
00B2 030C
                                          ! PUT PTRS !
00B4 E6FB      JR       Z,TC              ! REC NOTHING.... !
00B6 93F2      PUSH     @R15,R2
00B8 34CA      LDA      R10,R12(#GETBUF)
00BA 0000*
00BC 1FA0      CALL     @R10          ! GET RNGBUF ADR !
00BE 6FE0      LD       GETOUT(R14),R0
00C0 030E
00C2 2028      LDB      RL0,@R2       ! STO CHR FOR RTN !
00C4 97F2      POP      R2,@R15
```

```
                ! CHECK FOR NON-DISPLAY FROM LOAD_FILE !
00C6 0B09       CP      R9,#%AAAA
00C8 AAAA
00CA E603       JR      Z,NO_DISPLAY
00CC 34CA       LDA     R10,R12(SNDCHR)
00CE 01DE'
00D0 1FA0       CALL    @R10

                NO_DISPLAY:
00D2 9E08       RET
00D4            END CONRD


                GLOBAL
00D4            QUIT  PROCEDURE
                !*****************************************!
                 *                                       *
                 *  QUIT: TRANSMITS ALL CHR AND CR FROM  *
                 *        CONS TO MCZ; THEN RELAYS ALL   *
                 *        TO CONS FROM MCZ; AND ETC.     *
                 *                                       *
                 *****************************************!
                ENTRY
00D4 4DE8       CLR     MCZPUT(R14)
00D6 0310
00D8 4DE8       CLR     MCZGET(R14)
00DA 0312
00DC 65E0       SET     MFLAGS(R14),#TRPMDE  ! RESET BUF PTRS !
00DE 031C
                                     ! AND ENTER TRANSPARENT MODE !

                ! CONSOLE RECEIVE ROUTINE !
                PORTB:
00E0 61E0       LD      R0,GETOUT(R14)
00E2 03CE
00E4 4BE0       CP      R0,NXTPTR(R14)   ! CK FOR CONS INPUT !
00E6 030C
00E8 E60A       JR      Z,PORTA      ! NO, CK MCZ..... !

                ! PROCESS CONSOLE INPUT !
00EA 34CA       LDA     R10,R12(#GETBUF)
00EC 0000*
00EE 1FA0       CALL    @R10         ! GET RNGBUF ADR !
00F0 6FE0       LD      GETOUT(R14),R0  ! SET BEGIN PTR !
00F2 03CE
00F4 2028       LDB     RL0,@R2
00F6 34CA       LDA     R10,R12(#SNDMCZ)
00F8 0000*
00FA 1FA0       CALL    @R10         ! ECHO CHR TO MCZ !
00FC EEF1       JR      NZ,PORTB     ! CONTINUE UNTIL CR !
```

```
                ! MCZ RECEIVE ROUTINE !
                PORTA:
00FE 61E0         LD        R0,MCZGET(R14)
0100 0312
0102 4BE0         CP        R0,MCZPUT(R14)    ! CK FOR MCZ INPUT !
0104 0310
0106 E6EC         JR        Z,PORTB       ! NO, CK CONSOLE...!
0108 34CA         LDA       R10,R12(#GMCZAD)
010A 0000*
010C 1FA0         CALL      @R10          ! GET MCZBUF ADR !
010E 6FE0         LD        MCZGET(R14),R0
0110 0312
0112 2028         LDB       RL0,@R2   ! GET CHAR FROM MCZBUF !
0114 34CA         LDA       R10,R12(SNDCHR)
0116 01DE
0118 1FA0         CALL      @R10      ! OUTPUT CHR TO CONSOLE !
011A E8F1         JR        PORTA    ! CONTINUE TIL EMPTY !
011C            END QUIT

                GLOBAL
011C            SKPBLK  PROCEDURE
                !*****************************************
                 *                                       *
                 *  SKPBLK: SKIP OVER BLANKS TO NEXT     *
                 *          CHARACTER.                   *
                 *                                       *
                 *  REG USE:  RETURN  RL0 = 1ST NON-BLK  *
                 *                    CHAR AND Z IF =CR  *
                 *                                       *
                 *****************************************!
                ENTRY
                ! SKIP OVER BLANKS TO NEXT ARGUMENT !
011C DFE0         CALR      GETCHR
011E 9E06         RET       Z             ! GOT CR !
0120 0A08         CPB       RL0,#' '      ! CK FOR BLANK !
0122 2020
0124 E6FB         JR        Z,SKPBLK      ! YES,..... !
0126 9E08         RET                     ! GOT CHAR !

0128            END SKPBLK
```

```
                GLOBAL
0128            GETADR  PROCEDURE
                !*****************************************
                 *                                       *
                 *  GETADR: GETS NEXT ARGUMENT AND       *
                 *          CONVERTS TO HEX ADDRESS.     *
                 *                                       *
                 *  REG USE: INPUT  RL0 = 1ST CH OF ARG  *
                 *           RETURN  R3 = HEX ADR        *
                 *                  AND Z,C  IF CR ONLY  *
                 *                      Z,NC IF ARG,CR   *
                 *                     NZ,NC IF ARG,SP   *
                 *                                       *
                 *****************************************!
                ENTRY
                ! CK FOR CR ONLY !
0128 8D38          CLR       R3
012A 0A08          CPB       RL0,#CR          ! CK FOR CR !
012C 0D0D
012E EE02          JR        NZ,NOTCR
0130 8D81          SETFLG    C
0132 9E08          RET                  ! RETURN FOR CR ONLY !

                !CONVERT ASCII ADDRESS TO HEX ADDRESS !
                NOTCR:
0134 DFE2          CALR      CONVERT      !BYTE TO 4-BIT HEX !
0136 E70A          JR        C,REPERR     ! GOT BAD CHR !
0138 BEB8          RLDB      RL0,RL3
013A BE38          RLDB      RL0,RH3      !SHFT LEFT TO MSW !
013C DFF0          CALR      GETCHR       ! GET CHR FROM INTBUF!
013E 9E06          RET       Z
0140 0A08          CPB       RL0,#' '     !CK FOR SPACE !
0142 2020
0144 EEF7          JR        NZ,NOTCR     ! IF NOT, CONT.....!
0146 D016          CALR      SKPBLK       ! SKIP TO NEXT ARG !
0148 8D83          RESFLG    C
014A 9E08          RET                    ! SPACE AFTER ARG !

                REPERR:
014C 34CA          LDA       R10,R12(#EROR)
014E 0000*
0150 1EA8          JP        @R10
0152            END GETADR
```

```
                GLOBAL
0152            GETNXT  PROCEDURE
                !*****************************************
                 *                                       *
                 *  GETNXT: SKIP TO BEGINING OF NEXT     *
                 *          ARGUMENT IN COMMAND.         *
                 *                                       *
                 *  REG USE:  RETURN  RLØ = CHAR OR CR   *
                 *                    AND Z IF = CR      *
                 *                                       *
                 *****************************************!
                ENTRY
                ! SKIP OVER CURRENT ARGUMENT TO NEXT SPACE !
0152 DFFB         CALR      GETCHR
0154 9E06         RET       Z             ! RTN IF CH=CR !
0156 0A08         CPB       RLØ,#' '      ! FIND FIRST SPACE !
0158 2020
015A FEFB         JR        NZ,GETNXT
015C E8DF         JR        SKPBLK        ! NOW SKIP BLANKS !

015F            END GETNXT

                GLOBAL
015E            GETCHR  PROCEDURE
                !*****************************************
                 *                                       *
                 *  GETCHR: GETS NEXT CHR FROM INTBUF    *
                 *          AND INCREMENTS INTPTR.       *
                 *                                       *
                 *  REG USE: RETURN  RLØ = CHR           *
                 *                   AND Z IF CR         *
                 *                                       *
                 *****************************************!
                ENTRY
015E 93F2         PUSH      @R15,R2       ! SAVE WORK REG !
0160 61E2         LD        R2,INTPTR(R14)
0162 0304
0164 2028         LDB       RLØ,@R2       ! GET CHR !
0166 69E0         INC       INTPTR(R14),#1    ! INC PTR !
0168 0304
016A 0A08         CPB       RLØ,#CR        !CK FOR CR !
016C 0D0D
016E 97F2         POP       R2,@R15
0170 9E08         RET
0172            END GETCHR
```

```
                GLOBAL
0172            CONVERT  PROCEDURE
                !*************************************************
                 *                                               *
                 *  CONVERT: CONVERTS 8-BIT ASCII CHR            *
                 *           TO 4-BIT HEX VALUE. VALID           *
                 *           CHR IS 0-9 OR A-F; IF NOT           *
                 *           CHR, EXIT TO EXEC EROR.             *
                 *                                               *
                 *  REG USE: INPUT  RL0 = 8-BIT ASCII            *
                 *           RETURN RL0 = 4-BIT HEX IN           *
                 *                        LSW.                   *
                 *                                               *
                 *************************************************!
                ENTRY
                ! CHECK FOR VALID CHAR !
0172 0A08          CPB      RL0,#'0'  ! FILTER <'0' ASCII !
0174 3030
0176 9E07          RET      C         ! ERROR !
0178 0A08          CPB      RL0,#'9'+1   ! CK IF DIGIT '
017A 3A3A
017C E708          JR       C,NOFIX
017E 0A08          CPB      RL0,#'A'  ! FILTER <'A' ASCII !
0180 4141
0182 9E07          RET      C
0184 0A08          CPB      RL0,#'F'+1 !FILTER >'F' ASCII !
0186 4747
0188 EF06          JR       NC,RETSIG  !ERROR!
018A 0208          SUBB     RL0,#7     ! ALPHA ADJUST !
018C 0707
                NOFIX:
018E 0608          ANDB     RL0,#%0F   ! GET LOW NIBBLE !
0190 0F0F
0192 8D83          RESFLG   C
0194 9E08          RET                 ! RTN HEX VALUE !
                RETSIG:
0196 8D81          SETFLG   C
0198 9E08          RET                 ! RTN FOR BAD CHR !

019A            END CONVERT
```

```
                GLOBAL
                   PBUFNC  LABEL
019A               PRNTBF  PROCEDURE
                !*****************************************
                 *                                       *
                 *  PRNTBF: PRINT CONTENTS OF OUTBUF     *
                 *          TO CONS WITH CR AT END.      *
                 *                                       *
                 *  PBUFNC: PRINT BUFFER CONTENTS WITH   *
                 *          NO CR.                       *
                 *                                       *
                 *****************************************!
                ENTRY
                ! STORE CR IN OUTBUF !
019A 61E2          LD       R2,OUTPTR(R14)
019C 0306
019E 0C25          LDB      @R2,#CR
01A0 0D0D
01A2 69E0          INC      OUTPTR(R14),#1
01A4 0306

                PBUFNC:
01A6 76E1          LDA       R1,OUTBUF(R14)   ! LOAD ADR OF OUTBUF !
01A8 0080
                ! OUTPUT LOOP !
                PRNT:
01AA 2018          LDB      RL0,@R1          ! GET CHR !
01AC A910          INC      R1               ! INC INDEX !
01AE DFE9          CALR     SNDCHR           ! OUTPUT CHR !
01B0 E604          JR       Z,OUTLF          ! ?CHR = CR !
01B2 4BE1          CP       R1,OUTPTR(R14)     ! CK FOR END !
01B4 C306
01B6 E7F9          JR       C,PRNT           ! LOOP......!
01B8 E802          JR       FINI             ! FINISHED !

                ! ADD LF AFTER OUTPUT OF CR !
                OUTLF:
01BA C80A          LDB      RL0,#LF           ! OUTPUT LF !
01BC DFF0          CALR     SNDCHR

                ! FILL OUTBUF WITH BLANKS AND RESET OUTPTR !
                FINI:
01BE 76E3          LDA      R3,OUTBUF(R14)
01C0 0080
01C2 6FE3          LD       OUTPTR(R14),R3 ! RESET PTR !
01C4 0306
01C6 2100          LD       R0,#OUTSIZ/2-1 ! FILL CNT !
01C8 003F
01CA 4DE5          LD       OUTBUF(R14),#'  '
01CC 0080
```

```
01CE 2020
01D0 76E2       LDA       R2,OUTBUF'R14)
01D2 0080
01D4 A123       LD        R3,R2
01D6 A931       INC       R3,#2
01D8 BB21       LDI       @R3,@R2,R0     ! FILL OUTBUF !
01DA 0030
01DC 9E08       RET
01DE            END PRNTBF

                GLOBAL
01DE            SNDCHR  PROCEDURE
                !*************************************
                 *                                   *
                 *  SNDCHR: CK MONITOR FLAG WORD FOR *
                 *          OUTPUT STOP SIGNAL (OSTOP);*
                 *          IF NOT, SEND CHAR TO CONS. *
                 *                                   *
                 *  REG USE: INPUT   RL0= CHR        *
                 *           RETURN  RL0= CHR AND Z IF *
                 *                       CHR = CR.   *
                 *                                   *
                 *************************************!
                ENTRY
                ! WAIT FOR OUTPUT OK SIGNAL !
01DE 67E2       BIT       MFLAGS(R14),#OSTOP   ! CK FLAG !
01E0 031C
01E2 FEFD       JR        NZ,SNDCHR
                !OUTPUT CHAR TO TERMINAL !
01E4 3A04       INB       RH0,PORTBC   ! GET PORT STATUS !
01E6 FFE3
01E8 A600       BITB      RH0,#TXR      ! TRANS RDY? !
01EA E6F9       JR        Z,SNDCHR     ! NO, CONTINUE...!
01EC 3A86       OUTB      PORTBD,RL0   ! YES, OUTPUT CHR !
01EE FFE1
01F0 0A08       CPB       RL0,#CR
01F2 0D0D
01F4 9E08       RET
01F6            END SNDCHR
```

```
                GLOBAL
                   CONVB   LABEL
01F6               CONVW   PROCEDURE
                !*****************************************
                 *                                       *
                 *  CONVW: CONVERT INTERNAL WORD, 4-     *
                 *         4-BIT HEX VALUES TO FOUR      *
                 *         8-BIT ASCII REPRESENTATIONS   *
                 *         OF THE HEX VALUES.            *
                 *                                       *
                 *  CONVB: CONVERT INTERNAL BYTE HEX     *
                 *         VALUE TO ASCII CHARACTERS.    *
                 *                                       *
                 *  REG USE: INPUT  R5 = WORD/BYTE(S)    *
                 *                  R3 = CKSUM ACCUM     *
                 *           RETURN R3 = UPDATED ACCUM   *
                 *                       AND ASCII CHR   *
                 *                       IN OUTBUF       *
                 *                                       *
                 *****************************************!
                ENTRY
                ! CONVERT WORD !
01F6 A050          LDB       RH0,RH5      ! 1ST BYTE !
01F8 DFFF          CALR      NIBBLE
                ! CONVERT BYTE ENTRY POINT !
                CONVB:
01FA A0D0          LDB       RH0,RL5

                NIBBLE:
01FC BE08          RLDB      RL0,RH0      ! FIRST NIBBLE !
01FE DFFF          CALR      CONPUT
0200 BE08          RLDB      RL0,RH0      ! NEXT NIBBLE !
                !CONVERT NIBBLE TO ASCII CHAR AND STORE !
                CONPUT:
0202 0608          ANDB      RL0,#%0F     ! GET NIBBLE !
0204 0F0F
0206 808B          ADDB      RL3,RL0      ! UPDATE CKSUM !
0208 0A08          CPB       RL0,#%0A     ! 0-9? !
020A 0A0A
020C E702          JR        C, ASCII     ! YES... !
020E 0008          ADDB      RL0,#7       ! NO, CONVERT CHR !
0210 0707
                ASCII:
0212 0008          ADDB      RL0,#%30     ! CONVERT TO ASCII !
0214 3030

                ! STORE IF OUTBUF !
0216 93F1          PUSH      @R15,R1      ! SAVE R1 !
0218 61E1          LD        R1,OUTPTR(R14)
021A 0306
```

```
021C 2E18      LDB     @R1,RL0         ! STORE CHR !
021E 69E7      INC     OUTPTR(R14)
0220 0306
0222 97F1      POP     R1,@R15
0224 9E08      RET
0226       END CONVW

           END SUPPORT1
```

4. SUPPORT2 MODULE

```
Z80CCASM  2.02
LOC    OBJ CODE    STMT SOURCE STATEMENT

                       1 SUPPORT2 MODULE
                   $LISTON  $TTY
                   !*****************************************
                    *                                       *
                    *  SUPPORT TWO MODULE: MODULE TWO FOR   *
                    *        SECONDARY STORAGE INTERFACING  *
                    *                                       *
                    *****************************************!


                   CONSTANT

                      RXR       :=  2
                      TXR       :=  0
                      PAR       :=  7
                      PORTAD    :=  %FFD9
                      PORTBD    :=  %FFE1
                      PORTAC    :=  %FFDB
                      PORTBC    :=  %FFE3

                      IDPORT    :=  %FFCB
                      ICPORT    :=  %FFC9

                      TCMD   :=  %FFD2
                      TDTA   :=  %FFD0

                      BUS_LOCK  :=%FFF9
                      BUS_UNLOCK :=%FFF8
                      VINTR     :=  %(2)0001000200000000
                      VIBIT     :=  12
                      ESCAPE    :=  %13
                      BS        :=  %08
                      LINDEL    :=  %7F
                      CR        :=  %0D
                      LF        :=  %0A
                      TXOFCH    :=  %13
                      TXONCH    :=  %11
                      INSIZ     :=  128        ! INTBUF SIZE !
                      OUTSIZ    :=  128        ! OUTBUF SIZE !
                      RBSIZ     :=  256        ! RING BUFFER SIZE !
                   ! BIT POSITIONS IN MONITOR FLAG WORD !
                      TRPMDE    :=  0
                      ISTOP     :=  1
                      OSTOP     :=  2
```

```
                SNDMDE    :=  3
                LDMDE     :=  4
                ESC       :=  5
                TXMSK     := %6


             EXTERNAL
                EXEC      LABEL
                PBUFNC    LABEL
                NMI_RTN   LABEL
                CONVW     PROCEDURE
                PRNTBF    PROCEDURE
                BRKROU    LABEL
                NEWLNE    PROCEDURE
                GETBUF    PROCEDURE
                GETLNE    PROCEDURE
                LOADFL    PROCEDURE

             INTERNAL
             $SECTION DATA_DEC
             $ABS 0

0000            INTBUF    ARRAY [128 BYTE]
0080            OUTBUF    ARRAY [128 BYTE]
0100            RNGBUF    ARRAY [256 BYTE]
0200            MCZBUF    ARRAY [256 BYTE]

0300            BUFADR    WORD
0302            BUFSIZ    WORD

0304            INTPTR    WORD
0306            OUTPTR    WORD

0308            UNIMP     WORD
030A            BRKCNT    WORD

030C            NXTPTR    WORD
030E            GETOUT    WORD
0310            MCZPUT    WORD
0312            MCZGET    WORD
0314            BRKSTR    WORD
0316            BRKADR    WORD
0318            TMPSP     WORD
031A            TMPPSW    WORD

031C            MFLAGS    WORD

              ! USER REGISTER STORAGE !

031E            R0_       WORD
```

```
0320            R1_       WORD
0322            R2_       WORD
0324            R3_       WORD
0326            R4_       WORD
0328            R5_       WORD
032A            R6_       WORD
032C            R7_       WORD
032E            R8_       WORD
0330            R9_       WORD
0332            R10_      WORD
0334            R11_      WORD
0336            R12_      WORD
0338            R13_      WORD
033A            R14_      WORD
033C            R15_      WORD
033E            RPC_      WORD
0340            RFC_      WORD

0342            RETRY     WORD
0344            ADR_STR   WORD

                GLOBAL
                $SECTION SUPPORT2_PROC
                $REL 0

                GLOBAL
0000            NMI  PROCEDURE
                !*********************************************
                 *                                           *
                 *  NMI INT: NON-MASKABLE INTERRUPT          *
                 *           HANDLER FOR RETURN  TO          *
                 *           INITIALIZATION ENVIRONMENT      *
                 *           DUE TO ERROR IN MCZ LOAD.       *
                 *                                           *
                 *********************************************!
                ENTRY
0000 210F       ADD       R15,#6      ! RESTORE STACK PTR !
0002 0006
0004 76CA       LDA       R10,NMI_RTN(R12)
0006 0000*
0008 1EA8       JP        @R10
000A            END NMI
```

```
                GLOBAL
000A            LOAD_FILE  PROCEDURE
                !*****************************************
                 *                                       *
                 *  LOAD_FILE: LOADS MCZ FILENAME INTO   *
                 *             RINGBUFFER (CONS INPUT);  *
                 *             CONVERTS LOAD ADR (R3)    *
                 *             INTO ASCII  CHAR  AND     *
                 *             STORES IN RINGBUFFER.     *
                 *                                       *
                 *  REG USE: INPUT  R3 = HEX LOAD ADR    *
                 *                  R4 = FILENAME ADR    *
                 *                                       *
                 *****************************************!
                ENTRY
                ! LOAD FILENAME INTO RINGBUFFER !
000A 61E0          LD          R0,NXTPTR(R14)
000C 030C
000E A109          LD          R9,R0          ! SAVE RINGBUF OFFSET !
0010 76CA          LDA         R10,GETBUF(R12)
0012 0128
0014 1FA0          CALL        @R10           ! CONVERT TO ADR !
                ! RETURNS R2 = ADR !
0016 8D08          CLR         R0
0018 2048          LDB         RL0,@R4        ! 1ST BYTE=NO. CHR !
                                              ! IN FILENAME !
001A A940          INC         R4
001C 8109          ADD         R9,R0        ! UPDATE OFFSET !
001E 6FE9          LD          NXTPTR(R14),R9
0020 030C

0022 BA41          LDIRB       @R2,@R4,R0   ! ENTER FILENAME !
0024 0020

                !CONVERT LOAD ADDRESS TO ASCII !
0026 0B03          CP          R3,#%FFFE    ! CK FOR LOAD ADR !
0028 FFFE
002A E610          JR          Z,LD_FILE    ! USE IMAGED ADR !
002C A032          LDB         RH2,RH3      ! FIRST BYTE !
002E DFDA          CALR        HEX_TO_ASCII
0030 A124          LD          R4,R2        ! SAVE MSB'S !
0032 A0B2          LDB         RH2,RL3      ! GET SECOND BYTE !
0034 DFDD          CALR        HEX_TO_ASCII
0036 A125          LD          R5,R2        ! SAVE LSB'S !

                ! LOAD ADDRESS IN RINGBUFFER !
0038 61E0          LD          R0,NXTPTR(R14)  ! RINGBF OFFSET !
003A 030C
003C 76CA          LDA         R10,GETBUF(R12)
003E 0128
```

```
0040 1FA0      CALL      @R10              ! CONVERT TO ADR !
0042 2F24      LD        @R2,R4        ! ENTER MSB'S !
0044 3325      LD        R2(#2),R5         ! ENTER LSB'S !
0046 0002
0048 69E3      INC       NXTPTR(R14),#4  ! UPDATE PTR !
004A 030C

               LD_FILE:

               ! INSERT CARRIAGE RETURN !
004C 61E0      LD        R0,NXTPTR(R14)
004E 030C
0050 76CA      LDA       R10,GETBUF(R12)
0052 0128
0054 1FA0      CALL      @R10
0056 6FE0      LD        NXTPTR(R14),R0
0058 030C
005A 0C25      LDB       @R2,#%0D
005C 2D0D
               ! LOAD FILE WITH NO CONSOLE PROMPTING !
005E 2109      LD        R9,#%AAAA     ! NC_PROMPT SIGNAL !
0060 AAAA

               ! GET COMMAND !
0062 76CA      LDA       R10,GETLNE(R12)
0064 0000*
0066 1FA0      CALL      @R10

0068 7C05      EI        VI              ! ENABLE VECTORED INT !
006A 76CA      LDA       R10,LOADFL(R12)
006C 0000*
006E 1FA0      CALL      @R10            ! LOAD FILE !
0070 8D98      CLR       R9              ! DELETE SIGNAL !

0072 61E1      LD        R1,NXTPTR(R14)
0074 030C
0076 6FE1      LD        GETOUT(R14),R1   !RESET PTRS !
0078 030E
007A 9F08      RET

007C           END LOAD_FILE
```

```
007C            HEX_TO_ASCII  PROCEDURE
                !*****************************************
                 *                                       *
                 *  HEX_TO_ASCII: CONVERTS HEX ADR (TWO  *
                 *          4-BIT VALUES) TO ASCII (TWO  *
                 *          8-BIT VALUES).               *
                 *                                       *
                 *  REG USE: INPUT  RH2 = INPUT BYTE     *
                 *                                       *
                 *****************************************!
                ENTRY
007C BE2A          RLDB       RL2,RH2      ! CONVERT 1ST NIBBLE !
007E DFFD          CALR       HEX_ASCII
0080 A0A8          LDB        RL0,RL2      ! TEMP STORE BYTE !
0082 BE2A          RLDB       RL2,RH2      ! CONVERT 2ND NIBBLE !
0084 A082          LDB        RH2,RL0      ! RELOAD BYTE !

                HEX_ASCII:
0086 060A          ANDB       RL2,#%0F     ! GET NIBBLE !
0088 0F0F
008A 0A0A          CPB        RL2,#%0A     ! '0-9'? !
008C 0A0A
008E E702          JR         C,CONV_ASCII
0090 000A          ADDB       RL2,#7       ! CONVERT NUMERAL !
0092 0707

                CONV_ASCII:
0094 000A          ADDB       RL2,#%30     ! CONV TO ASCII !
0096 3030

0098 9E08          RET
009A            END HEX_TO_ASCII

                GLOBAL
009A            SNDMCZ  PROCEDURE
                !*****************************************
                 *                                       *
                 *  SNDMCZ: OUTPUT CHAR TO SERIAL PORT   *
                 *          ONE (MCZ SYS).               *
                 *                                       *
                 *  REG USE: INPUT  RL0 = CHAR           *
                 *           RETURN Z IF CHAR = CR       *
                 *                                       *
                 *****************************************!
                ENTRY
009A 3A04          INB        RH0,PORTAC   ! GET STATUS !
009C FFDB
009E A600          BITB       RH0,#TXR      ! TRANSMIT RDY? !
00A0 F6FC          JR         Z,SNDMCZ     ! NOT YET..... !
00A2 3A86          OUTB       PORTAD,RL0   ! YES, SND CHR !
```

```
00A4 FFD9
00A6 0A08       CPB     RL0,#CR
00A8 0D0D
00AA 9E08       RET
00AC            END SNDMCZ

                GLOBAL
00AC            SNDMSG  PROCEDURE
                !*****************************************
                 *                                       *
                 *  SNDMSG: SEND MSG SPECIFIED TO CONS   *
                 *          (PORT2). FIRST BYTE OF MSG   *
                 *          IS THE DECIMAL LENGTH IN     *
                 *          WORDS.                       *
                 *                                       *
                 *  REG USE: INPUT  R2 = MSG ADDR        *
                 *                                       *
                 *****************************************!
                ENTRY
00AC 34E1       LDA     R1,R14(#OUTBUF)
00AE 0080
00B0 8D08       CLR     R0
00B2 2028       LDB     RL0,@R2         ! GET BYTE COUNT !
00B4 8101       ADD     R1,R0
00B6 33E1       LD      R14(#OUTPTR),R1
00B8 0306
00BA A920       INC     R2          ! SETUP FOR TRANSFER !
00BC 34E1       LDA      R1,R14(#OUTBUF)
00BE 0080
00C0 BA21       LDIRB    @R1,@R2,R0  ! TRANSFER TO OUTBUF !
00C2 0010
00C4 34CA       LDA     R10,R12(#PBUFNC)
00C6 0000*
00C8 1EA8       JP      @R10        ! OUTPUT TO CONS !
00CA            END SNDMSG
```

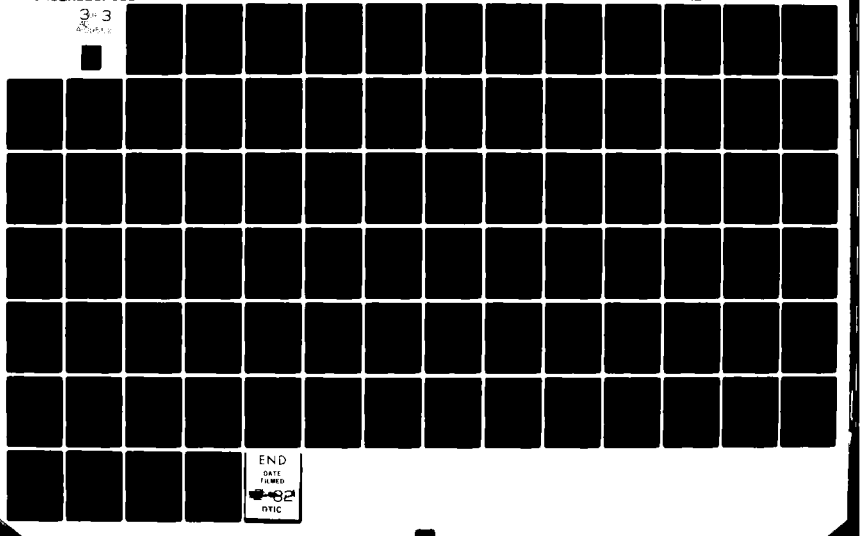
```
                GLOBAL
00CA            CONINT  PROCEDURE
                !*****************************************
                 *                                       *
                 *  CONINT: CONS (PORT2) INPUT INT HDLR  *
                 *          ROUTINE, WHICH GETS REC CHR  *
                 *          FROM USART. REC CHR IS CK    *
                 *          FOR TXOFCH OR TXONCH, AND    *
                 *          MFLAGS ADJUSTED ACCORDINGLY  *
                 *          TO SIGNAL PROCEDURES.        *
                 *                                       *
                 *****************************************!
                ENTRY
00CA 93F0          PUSH     @R15,R0
00CC 93F1          PUSH     @R15,R1
00CE 93F2          PUSH     @R15,R2      ! SAVE WORK REGS !
00D0 93FE          PUSH     @R15,R14
00D2 7D15          LDCTL    R1,PSAPOFF   ! DATA_AREA ADR !
00D4 211E          LD       R14,@R1

                ! GET CHAR AND CHECK FOR TXOFCH OR TXONCH !
00D6 3A94          INB      RL1,PORTBD   ! GET USART DATA !
00D8 FFE1
00DA A297          RESB     RL1,#PAR     ! CLR PARITY BIT !
00DC 67E0          BIT      MFLAGS(R14),#TRPMDE
00DE 031C
                                         ! TRANSPARENT MODE !
00E0 EE18          JR       NZ,PUTCHR    ! YES,.....!
00E2 0A09          CPB      RL1,#TXONCH   ! NO, CK FOR TXONCH!
00E4 1111
00E6 EE03          JR       NZ,AGAIN     ! NO,....!
00E8 63E2          RES      MFLAGS(R14),#OSTOP  ! RESET TO, !
00EA 031C
                                           ! RESUME OUTPUT !
00EC E818          JR       FINISH
                AGAIN:
00EE 0A09          CPB      RL1,#TXOFCH     ! CK FOR TXOFCH !
00F0 1313
00F2 EE03          JR       NZ,AGAIN2       ! NO,.... !
00F4 65E2          SET      MFLAGS(R14),#OSTOP  ! STOP OUTPUT !
00F6 031C
00F8 E812          JR       FINISH

                ! CHECK FOR ESCAPE CHARACTER !
                AGAIN2:
00FA 0A09          CPB      RL1,#ESCAPE
00FC 1B1B
00FE EE09          JR       NZ,PUTCHR    ! NO,.......!
0100 67E3          BIT      MFLAGS(R14),#SNDMDE ! YES, CK SND MDE!
0102 031C
```

```
0104 EE03     JR        NZ,ESCP       ! YES,.....!
0106 67E4     BIT       MFLAGS(R14),#LDMDE ! NO, CK LD MDE !
0108 031C
010A E603     JR        Z,PUTCHR      ! NO !
              ESCP:
010C 65E5     SET       MFLAGS(R14),#ESC  ! SET ESCAPE BIT !
010E 031C
0110 E806     JR        FINISH

              ! PRIMARY SAVE CHARACTER ROUTINE !
              PUTCHR:
0112 31E0     LD        R0,R14(#NXTPTR)
0114 030C
0116 DFF8     CALR      GETBUF        ! GET RNGBUF ADDR !
0118 33E0     LD        R14(#NXTPTR),R0
011A 030C
011C 2E29     LDB       @R2,RL1       ! PUT CHR IN RNGBUF !

              FINISH:
011E 97FE     POP       R14,@R15
0120 97F2     POP       R2,@R15
0122 97F1     POP       R1,@R15
0124 97F0     POP       R0,@R15     ! RESTORE WORK REGS !
0126 7B00     IRET
0128          END CONINT

              GLOBAL
0128          GETBUF  PROCEDURE
              !*****************************************
               *                                       *
               *  GETBUF: DETERMINES POSITION IN       *
               *          RINGBUFFER TO PUT OR GET     *
               *          NEXT CHAR.                   *
               *                                       *
               *  REG USE: INPUT  R0 = CURRENT INDEX   *
               *           RETURN R0 = NEW INDEX       *
               *                  R2 = ADR OF RNGBUF   *
               *                                       *
               *****************************************!
              ENTRY
0128 93FD     PUSH      @R15,R13
012A A102     LD        R2,R0
012C A900     INC       R0,#1         ! INC PTR !
012E 0B00     CP        R0,#RBSIZ     ! WRAP AROUND !
0130 0100
0132 EE01     JR        NZ,GB
0134 8D08     CLR       R0            ! RESET INDEX !
0136 34ED  GB:LDA       R13,R14(#RNGBUF)   ! NEW ADR !
0138 0100
013A 81D2     ADD       R2,R13
```

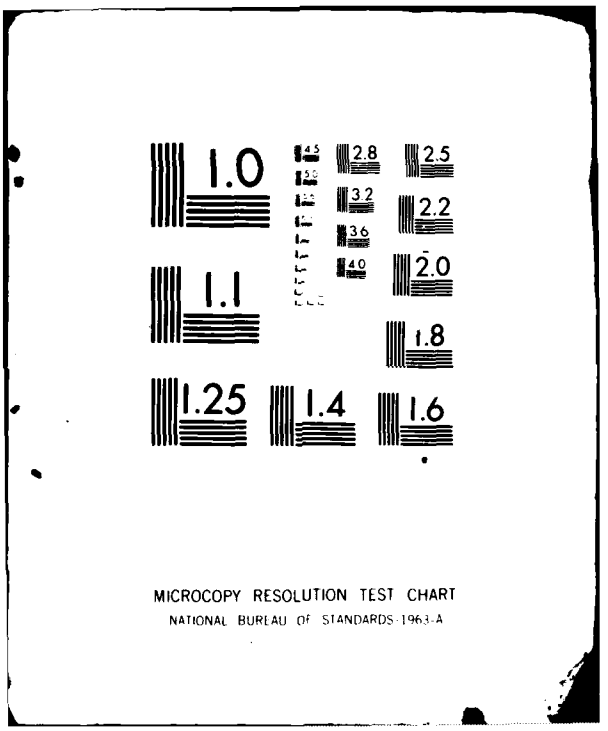
```
013C 97FD       POP     R13,@R15
013E 9F08       RET
0140            END GETBUF

                GLOBAL
0140            MCZHND  PROCEDURE
                !*****************************************
                *                                       *
                *  MCZHND: MCZ (SERIAL PORT1) INPUT     *
                *          INTERRUPT HANDLER ROUTINE    *
                *          WHICH GETS RECEIVED CHAR     *
                *          FROM USART, AND STORES IN    *
                *          MCZBUF.                      *
                *                                       *
                *****************************************!
                ENTRY
0140 93F0       PUSH    @R15,R0
0142 93F1       PUSH    @R15,R1
0144 93F2       PUSH    @R15,R2      ! SAVE WORK REGS !
0146 93FE       PUSH    @R15,R14
0148 7D15       LDCTL   R1,PSAPOFF
014A 211E       LD      R14,@R1      ! DATA_AREA ADR !
                ! GET CHAR FROM MCZ !
014C 3A94       INB     RL1,PORTAD   ! GET CHR !
014E FFD9
0150 A297       RESB    RL1,#PAR     ! RESET PARITY !
0152 31E0       LD      R0,R14(#MCZPUT)
0154 0310
0156 DFF8       CALR    GMCZAD       ! GET MCZBUF ADR !
0158 33E0       LD      R14(#MCZPUT),R0
015A 0310
015C 2E29       LDB     @R2,RL1      ! SAVE CHAR !

                !RESTORE WORK REGS !
015E 97FE       POP     R14,@R15
0160 97F2       POP     R2,@R15
0162 97F1       POP     R1,@R15
0164 97F0       POP     R0,@R15
0166 7B00       IRET
0168            END MCZHND
```

```
                GLOBAL
0168            GMCZAD  PROCEDURE
                !*****************************************
                 *                                       *
                 *  GMCZAD: GET NEXT ADR OF MCZ BUFFER   *
                 *          TO STORE OR GET CHARACTER.   *
                 *                                       *
                 *  REG USE: INPUT  R0 = PTR IN MCZBUF   *
                 *           RETURN R0 = NEW PTR IN BUF  *
                 *                  R2 = BGN OF MCZBUF   *
                 *                                       *
                 *****************************************!
                ENTRY
0168 93FD          PUSH    @R15,R13
016A A102          LD      R2,R0
016C A900          INC     R0,#1
016E 0B00          CP      R0,#RBSIZ       ! WRAP AROUND? !
0170 0100
0172 EE01          JR      NZ,GBZ
0174 8D08          CLR     R0              ! RESET OFFSET !
                GBZ:
0176 34FD          LDA     R13,R14(#MCZBUF)    ! GET ADR !
0178 0200
017A 81D2          ADD     R2,R13
017C 97FD          POP     R13,@R15
017E 9E08          RET
0180            END GMCZAD

                END SUPPORT2
```

AD-A109 552 NAVAL POSTGRADUATE SCHOOL MONTEREY CA F/G 9/2
SASS HARDWARE ARCHITECTURE AND DEVELOPMENTAL MONITOR.(U)
JUN 81 G S BAKER
UNCLASSIFIED NPS52-81-016 NL



END
DATE FILMED
-82
DTIC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

5. SUPPORT3 MODULE

```
Z8000ASM  2.02
LOC     OBJ CODE     STMT SOURCE STATEMENT

                        1 SUPPOPT3 MODULE
                    $LISTON  $TTY

                    !*****************************************
                     *                                       *
                     *  SUPPORT3 MODULE: MODULE THREE FOR    *
                     *       SECONDARY STORAGE PRIMITIVES    *
                     *       FUNCTIONS SUPPORT. STRICTLY     *
                     *       HARDWARE DEPENDENT; SHOULD      *
                     *       MEET STORAGE DEVICE REQUIRE-    *
                     *       MENTS FOR INTERFACING.          *
                     *                                       *
                     *  NOTE: DUPLICATE OF MONITOR LOAD_CMD  *
                     *        MODULE.                        *
                     *                                       *
                     *****************************************!


                    CONSTANT

                       RXR       :=  2
                       TXR       :=  Ø
                       PAR       :=  7
                       PORTAD    :=  %FFD9
                       PORTBD    :=  %FFE1
                       PORTAC    :=  %FFDB
                       PORTBC    :=  %FFE3

                       IDPORT    :=  %FFCB
                       ICPORT    :=  %FFC9

                       TCMD   :=  %FFD2
                       TDTA   :=  %FFDØ

                       BUS_LOCK :=%FFF9
                       BUS_UNLOCK :=%FFF8
                       VINTR     :=  %(2)0001000000000000
                       VIBIT     :=  12
                       ESCAPE    :=  %1B
                       BS        :=  %Ø9
                       LINDEL    :=  %7F
                       CR        :=  %ØD
                       LF        :=  %ØA
                       TXOFCH    :=  %13
```

```
                TXONCH    :=  %11
                INSIZ     :=  128        ! INTBUF SIZE !
                OUTSIZ    :=  128        ! OUTBUF SIZE !
                RBSIZ     :=  256        ! RING BUFFER SIZE !
             ! BIT POSITIONS IN MONITOR FLAG WORD !
                TRPMDE    :=  0
                ISTOP     :=  1
                OSTOP     :=  2
                SNDMDE    :=  3
                LDMDE     :=  4
                ESC       :=  5
                TXMSK     := %6

                COMDS     :=  12

             EXTERNAL     PRNTBF         PROCEDURE
             EXTERNAL     GETNXT         PROCEDURE
             EXTERNAL      EROR           LABEL
             EXTERNAL      SNDCHR         PROCEDURE
             EXTERNAL      GETADP         PROCEDURE
             EXTERNAL      GMCZAD         PROCEDURE
             EXTERNAL      SNDMCZ         PROCEDURE
             EXTERNAL      CONVERT        PROCEDURE
             EXTERNAL      PBUFNC         LABEL
             EXTERNAL     SNDMSG         PROCEDURE
             EXTERNAL     CONVW          PROCEDURE

             INTERNAL
             $SECTION DATA_DEC
             $ABS 0

0000            INTBUF    ARRAY [128 BYTE]
0080            OUTBUF    ARRAY [128 BYTE]
0100            RNGBUF    ARRAY [256 BYTE]
0200            MCZBUF    ARRAY [256 BYTE]

0300            BUFADR    WORD
0302            BUFSIZ    WORD

0304            INTPTR     WORD
0306            OUTPTR    WORD

0308            UNIMP     WORD
030A            BRKCNT    WORD

030C            NXTPTR    WORD
030E            GETOUT    WORD
0310            MCZPUT    WORD
0312            MCZGET    WORD
0314            BPKSTR    WORD
```

```
0316            BRKADR   WORD
0318            TMPSP    WORD
031A            TMPFCW   WORD

031C            MFLAGS   WORD

              ! USER REGISTER STORAGE !

031E            R0_      WORD
0320            R1_      WORD
0322            R2_      WORD
0324            R3_      WORD
0326            R4_      WORD
0328            R5_      WORD
032A            R6_      WORD
032C            R7_      WORD
032E            R8_      WORD
0330            R9_      WORD
0332            R10_     WORD
0334            R11_     WORD
0336            R12_     WORD
0338            R13_     WORD
033A            R14_     WORD
033C            R15_     WORD
033E            RPC_     WORD
0340            RFC_     WORD

0342            RETRY    WORD

0344            ADR_STR  WORD


              $SECTION LOAD_PROC
              $REL 0


              GLOBAL
0000          FNAME PROCEDURE
              !*****************************************
               *                                       *
               *   FNAME: RESETS TWO PTRS TO MCZBUF    *
               *          AND CHECKS FOR FILENAME.     *
               *                                       *
               *****************************************!
              ENTRY
0000 4DE8        CLR      MCZGET(R14)
0002 0312
0004 4DE8        CLR      MCZPUT(R14)        ! RESET BUFFER !
0006 0310
0008 34CA        LDA      R10,R12(#GETNXT)
```

```
000A 0000*
000C 1FA0       CALL    @R10            ! SKIP CMD ARG !
000E 0A09       CPB     RL0,#'A'
0010 4141
0012 E711       JR      C,DUN
0014 0A09       CPB     RL0,#'Z'+1
0016 5B5B
0018 EF0E       JR      NC,DUN          ! 1ST CHR IN (A..Z) !
001A 76CA       LDA     R10,GETNXT(R12)
001C 0000*
001E 1FA0       CALL    @R10            ! SKIP TO NEXT ARG !
0020 E607       JR      Z,NO_ADR        ! NO NEXT ARG !
0022 76CA       LDA     R10,GETADR(R12)
0024 0000*
0026 1FA0       CALL    @R10            ! GET USER SPECIFIED !
                                        ! ADDRESS !
0028 A13B       LD      R11,R3          ! SAVE USER ADR !
002A 6FE3       LD      ADR_STR(R14),R3
002C 0344
002E 9E08       RET

            NO_ADR:
0030 210B       LD      R11,#%FFFE      ! SIGNAL TO USE MCZ !
0032 FFFE
                                        ! ADDRESS !
0034 9E08       RET

            DUN:
0036 8D98       CLR     R9
0038 34CA       LDA     R10,R12(#EROR)
003A 0000*
003C 1EA8       JP      @R10            ! ERROR, RTN TO EXEC !
003E        END FNAME
```

```
                GLOBAL
003F            CMDPAS PROCEDURE
                !*****************************************!
                 *                                       *
                 *  CMDPAS: LOAD CMD PASSING MECHANISM   *
                 *          SENDS 'B;' PLUS CONS CMD     *
                 *  LINE TO MCZ AND CKS RESPONSES FOR    *
                 *  GOOD Z80 PROGRAM LOAD.               *
                 *                                       *
                 *  REG USE:  RETURN   NZ IF Z80 LOADED  *
                 *                     Z  IF NOT         *
                 *                                       *
                 *****************************************!
                ENTRY
003E 67E5          BIT      MFLAGS(R14),#ESC  ! CK FOR ESCAPE !
0040 031C
0042 E602          JR       Z,GCMD
0044 8D41          SETFLG   Z
0046 9E08          RET

                GCMD:
0048 C242          LDB      RH2,#'B'
004A CA3B          LDB      RL2,#';'
004C 6FE2          LD       OUTBUF(R14),R2     ! LOAD INIT 'B;' !
004E 0080
                                              ! FOR BRIEF MODE !
0050 76E2          LDA       R2,OUTBUF(R14)
0052 0080
0054 A921          INC      R2,#2
0056 76E1          LDA       R1,INTBUF(R14)
0058 0000
005A 2100          LD       R0,#%40          ! LD CMD IN OUTBUF !
005C 0040
005E BB11          LDIR     @R2,@R1,R0
0060 0020

0062 76E1          LDA      R1,OUTBUF(R14)
0064 0080
0066 0101          ADD      R1,#%80
0068 0080
006A 6FE1          LD       OUTPTR(R14),R1
006C 0306

006E DFB6          CALR     OUTSTM           ! OUTPUT BUFFER !
0070 DFE7          CALR     SKIPLN           ! SKIP MCZ ECHO !
0072 DFD4          CALR     MCZCOM           ! WAIT RESPONSE !
0074 0A09          CPB      RL1,#'B'
0076 4242
0078 EE02          JR       NZ,LDSTAT
007A DFEC          CALR     SKIPLN         ! SKIP MCZ ECHO !
```

```
007C DFD9     CALR     MCZCOM        ! WAIT RESPONSE !


              ! VERIFY LOAD STATUS !
              LDSTAT:
007E 0A09     CPB      RL1,#'9'      ! TEST LEGAL !
0080 3939
0082 F60E     JR       Z,RECACK      ! ACKNOWLEDGEMENTS !
0084 0A09     CPB      RL1,#'0'
0086 3030
0088 F60B     JR       Z,RECACK      ! REC GOOD ACK !
008A 0A09     CPB      RL1,#'7'
008C 3737
008E F608     JR       Z,RECACK

              ! NO ACKNOWLEDGEMENTS RECEIVED !
              ERMSG:
0090 DFD9     CALR     RECMSG        ! GET MCZ MSG !
0092 34CA     LDA      R10,R12(#SNDCHR)
0094 0000*
0096 1F40     CALL     @R10          ! SEND TO CONS !
0098 0A08     CPB      RL0,#LF
009A 0A0A
009C 9E06     RET      Z             ! DONE !
009E F8F8     JR       ERMSG

              ! ACKNOWLEDGE RECEIVED !
              RECACK:
00A0 8D43     RESFLG   Z             ! RETURN NZ !
00A2 9E08     RET
00A4          END CMDPAS

              GLOBAL
              SKPB     LABEL
00A4          SKIPLN   PROCEDURE
              !*****************************************
               *                                       *
               *  SKIPLN: SKIP RECEIVED LINE FROM      *
               *          MCZ; RETURN FIRST CHAR OF    *
               *          NEXT LINE.                   *
               *                                       *
               *  REG USE:  RETURN   RL1 = 1ST CHR     *
               *                     AND NZ IF ESC     *
               *                                       *
               *****************************************!
              ENTRY
00A4 DFE3     CALR     RECMSG        ! SKIP OVER LINE !
00A6 0A08     CPB      RL0,#CR       ! THRU CR,LF !
00A8 0D0D
00AA EEFC     JR       NZ,SKIPLN
```

```
                SKPB:
00AC 2101         LD        R1,#%3000      ! DELAY FACTOR !
00AE 3000
                ! MAIN LOOP FOR RECEIVING CHAR !
                LOOP1:
00B0 61E0         LD        R0,MCZGET(R14)
00B2 0312
00B4 4BE0         CP        R0,MCZPUT(R14)      ! TEST FOR REC CHR !
00B6 0310
00B8 EE03         JR        NZ,RECHR            ! YES,..... !
00BA AB10         DEC       R1,#1               ! NO, WAIT AWHILE !
00BC FEF9         JR        NZ,LOOP1
00BE 9E06         RET       Z                   ! FORCED EOL !

                RECHR:
00C0 DFFB         CALR      MCZCOM
00C2 0A09         CPB       RL1,#' '            ! CK 1ST=PRNT CHR !
00C4 2020
00C6 9E0D         RET       PL
00C8 DFF5         CALR      RECMSG
00CA E8F0         JR        SKPB
00CC            END SKIPLN

00CC            MCZCOM PROCEDURE
                !*****************************************
                 *                                       *
                 *  MCZCOM: LOOPS WAITING FOR RECEIVE    *
                 *          CHAR FROM MCZ BY SEEING IF   *
                 *          MCZBUF GETS CHAR. DOES       *
                 *          ADVANCE POINTER.             *
                 *                                       *
                 *  REG USE: RETURN  RL1 = CHR           *
                 *                                       *
                 *****************************************!
                ENTRY
00CC 61E0         LD        R0,MCZGET(R14)     ! CHECK MCZBUF !
00CE 0312
00D0 4BE0         CP        R0,MCZPUT(R14)     ! POINTERS !
00D2 0310
00D4 E6FB         JR        Z,MCZCOM      ! WAIT.....!
00D6 34CA         LDA       R10,R12(#GMCZAD)
00D8 0000*
00DA 1FA0         CALL      @R10          ! GET CHAR FROM BUF !
00DC 2029         LDB       RL1,@R2
00DE 9E08         RET
00E0            END MCZCOM
```

```
00E0            RECMSG PROCEDURE
                !*****************************************
                 *                                       *
                 *  RECMSG: LOOPS WAITING FOR REC CHR    *
                 *          FROM MCZ. GETS CHAR AND      *
                 *          DO NOT ADVANCE BUF PTR.      *
                 *                                       *
                 *  REG USE:  RETURNS   RL0 = CHR        *
                 *                                       *
                 *****************************************!
                ENTRY
00E0 61E0          LD       R0,MCZGET(R14)
00E2 0312
00E4 1BE0          CP       R0,MCZPUT(R14)    ! CK FOR REC !
00E6 0310
00E8 E6F8          JR       Z, RECMSG     ! WAIT..... !
00EA 34CA          LDA      R10,R12(#GMCZAD)
00EC 0000*
00EE 1FA0          CALL     @R10          ! GET 1ST CHAR !
00F0 6FE0          LD       MCZGET(R14),R0   ! RESTORE PTR !
00F2 0312
00F4 2028          LDB      RL2,@R2       ! RTN CHAR !
00F6 9E08          RET
00F8            END RECMSG

                GLOBAL
                OUTSTM    LABEL
00F8            OUTLNE    PROCEDURE
                !*****************************************
                 *                                       *
                 *  OUTLNE: OUTPUTS A LINE OF CHAR FROM  *
                 *          OUTBUF TO MCZ WITH CR AT     *
                 *          END.                         *
                 *                                       *
                 *  OUTSTM: OUTPUTS A LINE OF CHAR W/CR  *
                 *                                       *
                 *****************************************!
                ENTRY
00F8 61E2          LD       R2,OUTPTR(R14)
00FA 0306
00FC 0C25          LDB      @R2,#CR        !STORE CR IN BUF !
00FE 0D0D
0100 69E0          INC      OUTPTR(R14),#1      ! INC PTR !
0102 0306

                ! NO CR ENTRY POINT !
                OUTSTM:
0104 76F1          LDA       R1,OUTBUF(R14)
0106 0080
                ! MAIN LOOP !
```

```
                OVRAGN:
0108 2018           LDB     RL0,@R1
010A A910           INC     R1
010C 34CA           LDA     R10,R12(#SNDMCZ)
010E 0000*
0110 1FA0           CALL    @R10            ! SND CHR TO MCZ !
0112 F603           JR      Z,FINIS
0114 4BE1           CP      R1,OUTPTR(R14)
0116 0306
0118 E7F7           JR      C,OVRAGN        ! CK IF BUF EMPTY !

                ! FINISHED, RESET OUTPTR(R14) AND BLANK OUTBUF !
                FINIS:
011A 76E2           LDA     R2,OUTBUF(R14) ! RESET POINTER !
011C 0080
011E 6FE2           LD      OUTPTR(R14),R2
0120 0306
0122 2100           LD      R0,#OUTSIZ/2
0124 0040
0126 AB00           DEC     R0,#1           ! SET COUNT !
0128 4DE5           LD      OUTBUF(R14),#'  '    ! LOAD COUNT !
012A 0080
012C 2020
012E 76E2           LDA     R2,OUTBUF(R14)
0130 0080
0132 A121           LD      R1,R2
0134 A911           INC     R1,#2
0136 BB21           LDIR    @R1,@R2,R0      ! CLR BUFFER !
0138 0010
013A 9E08           RET
013C            END OUTLNE
```

```
                ABORTM    LABEL
                GODPAK    LABEL
0130            BADPAK    PROCEDURE
                !*****************************************
                 *                                       *
                 *  BADPAK: SENDS RESEND SIGNAL ('7')    *
                 *          TO MCZ FOR BAD CKSUM OR REC  *
                 *          NON-ASCII CHR.               *
                 *                                       *
                 *  ABORTM: SENDS ABORT SIGNAL ('9')     *
                 *          WHEN USER SELECTED.          *
                 *                                       *
                 *  GODPAK: SENDS ACK SIGNAL ('0') FOR   *
                 *          RECEIPT OF GOOD PACKET.      *
                 *                                       *
                 *****************************************!
                ENTRY
0130 C837          LDB      RL0,#'7'        ! LD RESEND SIG !
0132 E803          JR       OUTALL
                ABORTM:
0134 C839          LDB      RL0,#'9'        ! LD ABORT SIG !
0136 E801          JR       OUTALL
                GODPAK:
0138 C830          LDB      RL0,#'0'        ! LD REC OK SIG !
                OUTALL:
013A 6FE8          LDB      OUTBUF(R14),RL0
013C 0080
013E 76ED          LDA      R13,OUTBUF(R14)
0140 0080
0142 A9D0          INC      R13,#1
0144 6FED          LD       OUTPTR(R14),R13
0146 0306
0148 D02F          CALR     OUTLNE          ! SEND MCZ SYSTEM !
014A D05A          CALR     SKIPLN          ! SKIP ECHO !
014C 9E08          RET
014E            END BADPAK
```

```
                GLOBAL
0158            GETACK   PROCEDURE
                !*****************************************
                 *                                       *
                 *  GETACK: RECEIVE AND INTERPRET ACK    *
                 *          FROM MCZ. GOOD ACK = '0'     *
                 *                    BAD  ACK = '7'     *
                 *                    ABORT    = '9'     *
                 *                                       *
                 *  REG USE:  RETURN  Z,NC IF GOOD ACK   *
                 *                    NZ,NC IF BAD ACK   *
                 *                    NZ,C IF ABORT      *
                 *                                       *
                 *****************************************!
                ENTRY
0158 D048          CALR     MCZCOM          ! GET CHR !
015C 0A09          CPB      RL1,#'0'        ! CK FOR ACK !
015F 3030
0160 FE04          JR       NZ,NACK         ! NO..... !
0162 D060          CALR     SKIPLN          ! YES, REC ACK !
0164 8D41          SETFLG   Z
0166 8D83          RESFLG   C
0168 9E08          RET

                ! CK FOR '7' AND '9' NON-ACKNOWLEDGEMENTS !
                NACK:
016A 0A09          CPB      RL1,#'7'        ! CK FOR RESEND !
016C 3737
016E FE04          JR       NZ,ABRT         ! NO.... !
0170 D067          CALR     SKIPLN
0172 8D43          RESFLG   Z
0174 8D83          RESFLG   C
0176 9E09          RET

                ! CHECK FOR ABORT !
                ABRT:
0178 0A09          CPB      RL1,#'9'
017A 3939
017C F602          JR       Z,ENDIT         ! YES, ABORT... !
017E D050          CALR     RECMSG          ! GET ANOTHER CHR !
0180 F9FC          JR       GETACK          ! TRY AGAIN.... !

                ENDIT:
0182 D070          CALR     SKIPLN
0184 8D43          RESFLG   Z
0186 8D81          SETFLG   C
0188 9E09          RET
018A            END GETACK
```

```
018A                LINRCT PROCEDURE
                    !*****************************************
                     *                                       *
                     *  LINRCT: RECEIVES LINE OF CHAR FROM   *
                     *          MCZ AFTER RECEIPT OF '/',    *
                     *          AND STORES IN INTBUF, ADDING*
                     *          CR AT END AND FILTERING OUT  *
                     *          CONTROL CHARACTERS. (<20H)   *
                     *          (TRUNCATES AFTER 80 CHAR)    *
                     *                                       *
                     *****************************************!
                    ENTRY
                    ! WAIT FOR ASCII / !
018A D256           CALR    RECMSG
018C 0A08           CPB     RL0,#'/'
018E 2F2F
0190 FEFC           JR      NZ,LINRCT     ! WAIT !
                    ! BEGIN STORING CHARACTERS !
0192 76E4           LDA     R4,INTBUF(R14)
0194 0000
0196 CB50           LDB     RL3,#80       !SET LINE LENGTH !
                    ! STORE CHAR IN INTBUF !
                    LOPSTR:
0198 D05D           CALR    RECMSG        ! GET CHAR !
019A 2E48           LDB     @R4,RL0       ! STORE !
019C 0A08           CPB     RL0,#CR       ! CK FOR END !
019E 0D0D
01A0 EE02           JR      NZ,SKPSOM     ! GOT CHAR.. !
01A2 D07C           CALR    SKPB
01A4 9E08           RET

                    !CONTROL CHAR FILTERED AND DEC LINE COUNTER !
                    SKPSOM:
01A6 0A08           CPB     RL0,#' '
01A8 2020
01AA F7F6           JR      C,LOPSTR
01AC A940           INC     R4,#1         ! GOOD CHAR !
01AE FB0C           DBJNZ   RL3,LOPSTR    ! DEC COUNT !

                    !TRUNCATE, TOO MANY CHAR !
                    LOPOVR:
01B0 D069           CALR    RECMSG
01B2 0A08           CPB     RL0,#CR       ! LOOK FOR CR !
01B4 0D0D
01B6 FEFC           JR      NZ,LOPOVR
01B8 76ED           LDA     R13,INTBUF(R14)
01BA 0000
01BC 010D           ADD     R13,#80
01BE 0050
01C0 2ED8           LDB     @R13,RL0
```

```
01C2 9F09       RET
01C4            END LINRCT

01C4            UNPACK   PROCEDURE
                !*****************************************!
                 *                                       *
                 *  UNPACK: UNPACKS RECEIVED PACKETS     *
                 *          FROM MCZ IN INTBUF AND       *
                 *          LOADS IN SPECIFIED MEMORY    *
                 *          AREA. ASCII CHAR ARE CON-    *
                 *          VERTED TO HEX VALUES.        *
                 *                                       *
                 *  REG USE: INPUT  RH3 = #BYTE DATA     *
                 *****************************************!
                ENTRY
01C4 A03C       LDB     RL4,RH3         ! SAVE COUNT !
01C6 FFDF       CALR    CONVAD          ! CONV START ADR !

                ! CHECK FOR USER ENTERED ADDR FOR LOAD !
01C8 0B0B       CP      R11,#%FFFE
01CA FFFE
01CC E601       JR      Z,USE_MCZADR
01CE A1B1       LD      R1,R11          ! USER SPECIFIED !

                USE_MCZADR:
01D0 76E2       LDA     R2,INTBUF(R14)
01D2 0000
01D4 A927       INC     R2,#8

                CANDS:
01D6 FFF9       CALR    TRNHEX      ! CONVERT 2-ASCII CHR !
01D8 2E18       LDB     @R1,RL0     ! STORE IN MEM !
01DA A910       INC     R1,#1
01DC FC04       DBJNZ   RL4,CANDS   ! CONV AND STORE ALL !

                ! UPDATE USER SPECIFIED ADDRESS !
01DE 0B0B       CP      R11,#%FFFE
01E0 FFFE
01E2 E601       JR      Z,NO_UPDATE   ! USE MCZ ADR !
01E4 A11B       LD      R11,R1      ! UPDATE USER ADR !

                NO_UPDATE:
01E6 9F08       RET

01E8            END UNPACK
```

```
01F8          TRNHEX PROCEDURE
              !*********************************************
               *                                           *
               *  TRNHEX: CONVERTS TWO ASCII CHAR FRM      *
               *          INTBUF TO TWO 4-BIT HEX #        *
               *          AND ADD TO CKSUM.                *
               *                                           *
               *  REG USE: INPUT  R2 = PTR TO 1ST CHR      *
               *                  RL3= CKSUM ACCUM         *
               *           RETURN R2 = UPDATE PTR          *
               *                  RL3= UPDATED ACCUM       *
               *                  RL0= HEX VALUE           *
               *                  AND C IF NON-ASCII       *
               *                     NC IF ALL GOOD        *
               *                                           *
               *********************************************!
              ENTRY
01F8 DFF6        CALR      ATOHEX          ! CONVERT 1ST CHR !
01FA 9F07        RET       C
01FC 808B        ADDB      RL3,RL0         ! ADD TO CKSUM !
01FE B3C9        SLA       R2,#12          ! MOVE TO H NIBBLE !
0200 000C
0202 DFF3        CALR      ATOHEX          ! CONVERT 2ND CHR !
0204 9F07        RET       C
0206 808B        ADDB      RL3,RL0
0208 8408        ORB       RL0,RH0         ! COMBINE NIBBLES !
020A 8D83        RESFLG    C
020C 9E08        RET
020E          END TRNHEX

01FF          ATOHEX PROCEDURE
              !*********************************************
               *                                           *
               *  ATOHEX: CONVERTS ONE ASCII CHAR TO       *
               *          4-BIT HEX NIBBLE.                *
               *                                           *
               *  REG USE: INPUT  R2 = PTR TO CHR          *
               *           RETURN R2 = PTR + 1             *
               *                  RL0= HEX NIBBLE          *
               *                                           *
               *********************************************!
              ENTRY
01FF 2028        LDB       RL0,@R2
0200 A920        INC       R2,#1           ! INC PTR !
0202 34CA        LDA       R10,R12(#CONVERT)
0204 0000*
0206 1FA0        CALL      @R10
0208 9E08        RET
020A          END ATOHEX
```

```
020A          CONVAD PROCEDURE
              !*****************************************
               *                                       *
               *  CONVAD: CONVERTS STARTING ADDRESS    *
               *          OF PACKET DATA TO HEX #.     *
               *                                       *
               *  REG USE: RETURN   R1 = ADDRESS(HEX)  *
               *                                       *
               *****************************************!
              ENTRY
020A 76E2        LDA       R2,INTBUF(R14)
020C 0000
020E D014        CALR      TRNHEX
0210 A091        LDB       RH1,RL0       ! STORE 1ST BYTE !
0212 D016        CALR      TRNHEX
0214 A089        LDB       RL1,RL0       ! STORE 2ND BYTE !
0216 9E08        RET
0218          END CONVAD

0218          CHKPAK   PROCEDURE
              !*****************************************
               *                                       *
               *  CHKPAK: CK RECEIVED MCZ PAC CKSUM    *
               *          AGAINST ACCUMULATED HEX      *
               *          VALUE CKSUM AFTER ASCII-TO-  *
               *          HEX CONVERSION.              *
               *                                       *
               *  REG USE:  RETURN  RH3 = BYTE COUNT   *
               *                    AND C IF BAD OR    *
               *                    NON-ASCII.         *
               *                                       *
               *****************************************!
              ENTRY
0218 76E2        LDA       R2,INTBUF(R14)
021A 0000
021C C303        LDB       RH3,#3
021E DFF9        CALR      CHKSUM          ! CK 1ST CKSUM !
0220 9E07        RET       C               ! PAD CK !
0222 8C34        TESTB     RH3
0224 9E06        RET       Z               ! NO DATA !
0226 93F3        PUSH      @R15,R3         ! SAVE BYTE COUNT !
0228 DFFE        CALR      CHKSUM          ! CK 2ND CKSUM !
022A 97F3        POP       R3,@R15
022C 9E08        RET
022E          END CHKPAK
```

```
022F          CHKSUM   PROCEDURE
              !*****************************************
               *                                       *
               *  CHKSUM: CONVERTS ALL REC ASCII CHR   *
               *          IN PAC TO HEX AND ACCUM NEW  *
               *          CKSUM. COMPARE CKSUMS AND    *
               *          REPORT DIFFERENCES.          *
               *                                       *
               *  REG USE: INPUT  R2 = PTR TO PAC      *
               *                  RH3= # CHR PAIRS     *
               *           RETURN RH3= BYTE COUNT      *
               *                  RL3= NEW CKSUM       *
               *                  RH3= REC CKSUM       *
               *                  AND C IF BAD OR      *
               *                  NON-ASCII REC        *
               *                                       *
               *****************************************!
              ENTRY
0230 8CF8        CLRB     RL3          ! RESET CKSUM !
0232 D025     AB:CALR     TRNHEX       ! CONVERT PAIRS !
0234 9E07        RET      C
0236 F303        DBJNZ    RH3,AB       ! CONTINUE..... !
0238 A083        LDB      RH3,RL0
023A 93F3        PUSH     @R15,R3      ! SAVE BYTE CNT !
023C D02A        CALR     TRNHEX       ! CONVERT NEXT TWO !
023E 97F3        POP      R3,@R15
0240 9F07        RET      C
0242 8AB8        CPB      RL0,RL3      ! COMPARE CKSUMS !
0244 9E06        RET      Z            ! GOOD CK... !
0246 8D81        SETFLG   C            ! BAD CKSUM !
0248 9E08        RET
024A          END CHKSUM
```

```
                GLOBAL
0248            LOADFL PROCEDURE
                !*****************************************
                 *                                       *
                 *  LOADFL: RECEIVES PACKET FROM MCZ IN  *
                 *          FOLLOWING FORMAT:            *
                 *                                       *
                 *  <ADR><CNT><CKS1><DTA>...<DTA><CKS2>  *
                 *                                       *
                 *  ADR = START ASR IN Z8000 MEM         *
                 *  CNT = # DATA WORDS                   *
                 *  CKS1= CKSUM OF <ADR> + <CNT>         *
                 *  <DTA>...<DTA> = 30 DATA WORDS        *
                 *  CKS2= CKSUM OF DATA HEX VALUES       *
                 *                                       *
                 *  PROCEDURE VERIFIES CKSUMS BEFORE     *
                 *  STORING DATA IN Z8000 MEM. PACKETS   *
                 *  ARE ACK FOR WITH: '0' = GOOD         *
                 *                    '7' = RESEND       *
                 *                    '9' = ABORT        *
                 *  IF REC '//' FROM MCZ, ECHOS WHAT     *
                 *  REC NEXT TO CONSOLE AND ABORT.       *
                 *                                       *
                 *****************************************!

                ENTRY
0248 D125          CALR     FNAME          ! CK FILENAME !
024A 65E4          SET      MFLAGS(R14),#LDMDE !SIGNAL LOAD IN !
024C 031C
                                             !PROGRESS!
024E D109          CALR     CMDPAS         ! SND CMD TO MCZ !
0250 9F06          RET      Z              ! Z80 PROG NO LOAD !
                RECLOP:
0252 D065          CALR     LINRCT         ! GET PACKET !
0254 76F2          LDA      R2,INTBUF(R14)
0256 0000
0258 2028          LDB      RL0,@R2
025A 0408          CPB      RL0,#'/'       ! CK FOR '//' !
025C 2F2F
025E EE10          JR       NZ,CONTIN      !NO, CONTINUE...!
0260 76E1          LDA      R1,OUTBUF(R14)   !YES,!
0262 0080
0264 2103          LD       R3,#%20
0266 0020
0268 BB21          LDIR     @R1,@R2,R3     !ERROR MSG SETUP !
026A 0310
026C 76E1          LDA      R1,OUTBUF(R14)
026E 0080
0270 0101          ADD      R1,#%20
0272 0020
```

```
0274 6FE1      LD       OUTPTR(R14),R1 !SET OUTPTR !
0276 0306

0278 34CA      LDA      R10,R12(#PBUFNC)
027A 0000*
027C 1FA0      CALL     @R10
027E 9F08      RET

               CONTIN:
0280 67E5         BIT      MFLAGS(R14),#ESC   ' CK FOR ABORT !
0282 031C
0284 FE34         JR       NZ,ABT          ! YES, ABORT...!
0286 D039         CALR     CHKPAK          ! CK CKSUMS !
0289 FF02         JR       NC,GDLD         ! GOOD LOAD !
028A D0A8         CALR     BADPAK          ! SEND NON-ACK !
028C E8F2         JR       RECLOP          ! TRY AGAIN !

               ! CHECK FOR LAST PACKET AND PRINT <ENT ADR> !
               GDLD:
028E 8CR8         CLRB     RL3
0290 8138         ADD      R2,R3    ! ACCUM NUMBER BYTES !
                                    ! OF TRANSFER !

0292 9C34         TESTB    RH3             ! CK COUNT=0 !
0294 EE28         JR       NZ,STOR         ! OK, BEGIN STR !
0296 D0AA         CALR     GODPAK          ! SEND GOOD ACK !
0298 54F0         LDL      RR0,INTBUF(R14)
029A 0000
029C 76FD         LDA      R13,OUTBUF(R14)
029E 0080
02A0 010D         ADD      R13,#%0C
02A2 000C

               ! CHECK FOR USER SPECIFIED ADDR !
02A4 0B09         CP       R9,#%AAAA
02A6 AAAA
02A8 E61D         JR       Z,END_LOAD    ! NO ECHO TO CONS !
02AA 0B0B         CP       R11,#%FFFE    ! CK FOR LOAD ADR !
02AC FFFE
02AE F608         JR       Z,SAME_ADR    ! USE MCZ ADR !
02B0 6FED         LD       OUTPTR(R14),R13  ! SET OUTBUF ADR !
02B2 0306
02B4 61E5         LD       R5,ADR_STR(R14) ! GET USER ADR !
02B6 0344
02B8 76CA         LDA      R10,CONVW(R12)
02BA 0000*
02BC 1FA0         CALL     @R10         ! CONVERT TO ASCII AND !
                                        ! AND STORE IN OUTBUF !
02BE E801         JR       FIN_BUF
               SAME_ADR:
```

```
02C0 1DD0       LDL     @R13,RR0

                FIN_BUF:
02C2 3402       LDA     R2,ENTADR       !LOAD ENTRY LABEL!
02C4 0048
02C6 76F1       LDA     R1,OUTBUF(R14)
02C8 0060
02CA 2100       LD      R0,#6
02CC 0006
02CE BB21       LDIR    @R1,@R2,R0
02D0 0010
02D2 76ED       LDA     R13,OUTBUF(R14)
02D4 0080
02D6 010D       ADD     R13,#%10
02D8 0010
02DA 6FFD       LD      OUTPTR(R14),R13
02DC 0306
02DE 34CA       LDA     R10,R12(#PRNTBF)
02E0 0000*
02E2 1FA0       CALL    @R10            ! PRINT MESSAGE !
                END_LOAD:
02E4 9E08       RET

                STOR:
02E6 D06F       CALR    CONVAD
02E8 D0D3       CALR    GODPAK          ! SEND ACK !
02EA D094       CALR    UNPACK          ! UNPACK AND STORE !
02EC E8B2       JR      RECLOP          ! CONTINUE....!

                ABT:
02EE 3402       LDA     R2,EMSG
02F0 020A
02F2 34CA       LDA     R10,R12(#SNDMSG)
02F4 0000*
02F6 1FA0       CALL    @R10            ! SEND MESSAGE !
02F8 D0DD       CALR    ABORTM          ! SEND ABORT !
02FA 9F08       RET

02FC            END LOADFL

                EMSG:
02FC 07         BVAL    7
02FE 2F41       WVAL    '/A'
0300 424F       WVAL    'BO'
0302 5254       WVAL    'RT'
0304 0D         BVAL    %0D
                ENTADR:
0306 454E       WVAL    'EN'
0308 5452       WVAL    'TR'
030A 5920       WVAL    'Y '
```

```
03CC 504F      WVAL  'PO'
03CE 494E      WVAL  'IN'
0310 5420      WVAL  'T '

               END SUPPORT3
```

A. BOOTSTRAP PROGRAM LISTING

1. BOOTSTRAP MODULE

```
Z8000ASM  2.02
LOC     OBJ CODE     STMT SOURCE STATEMENT

                        1 BOOTSTRP MODULE
                    $LISTON  $TTY

                    CONSTANT

                       RTR       :=  2
                       TXR       :=  3
                       PAR       :=  7
                       PORTAD    :=  %FFD9
                       PORTBD    :=  %FFE1
                       PORTAC    :=  %FFDB
                       PORTBC    :=  %FFE3

                       IDPORT    :=  %FFCB
                       ICPORT    :=  %FFC9

                       TCMD   :=  %FFD2
                       TDTA   :=  %FFD0

                       BUS_LOCK :=%FFF9
                       BUS_UNLOCK :=%FFF8
                       VINTR     :=  %(2)0001000000000000
                       VIBIT     :=  12
                       ESCAPE    :=  %1B
                       MAX_CPU   :=  8
                       BS        :=  %08
                       LINDEL    :=  %7F
                       CR        :=  %0D
                       LF        :=  %0A
                       TXOFCH    :=  %13
                       TXONCH    :=  %11
                       INSIZ     :=  128        ! INTBUF SIZE !
                       OUTSIZ    :=  128        ! OUTBUF SIZE !
                       RBSIZ     :=  256        ! RING BUFFER SIZE !
                    ! BIT POSITIONS IN MONITOR FLAG WORD !
                       TRPMDE    :=  0
                       ISTOP     :=  1
                       OSTOP     :=  2
```

```
            SNDMDE    :=  3
            LDMDE     :=  4
            ESC       :=  5
            TXMSK     :=  %6

         TYPE
            MESSAGE    ARRAY[3 WORD]
            SWITCH     ARRAY[2 WORD]
            MEM_ARRAY          ARRAY[32 WORD]
            CPU_ENTRY        RECORD [
                     SIGNAL          WORD
                     CPU_ID          WORD
                     MSG_BLK         MESSAGE
                     MEM_MAP         MEM_ARRAY]
            ID_ARRAY           ARRAY[MAX_CPU WORD]
            ENTRY_ARRAY        ARRAY[MAX_CPU CPU_ENTRY]

         INTERNAL
         $SECTION TABLE1_DATA
         $ABS 0

0000        CONFIG_TABLE    RECORD [
                     RW_PATTERN      WORD
                     CPU_NUM         WORD
                     NORM_RW_PAT     WORD
                     NORM_CPU_CNT    WORD
                     TABLE_LOCK      WORD
                     CPU_CNT         WORD
                     CPU_LIST        ENTRY_ARRAY]

         $SECTION TABLE2_DATA
         $ABS 0

0000        MIN_CONFIG_TBL  RECORD [
                     LOG_TO_PHYS     ID_ARRAY
                     CONF_MEM_MAP    MEM_ARRAY]

         $SECTION PSA_DATA
         $ABS 0
0000        PSA    RECORD [
                     DATA_AREA      WORD
                     CODE_AREA        WORD
                     UNIMP_INST     SWITCH
                     PRIV_INST      SWITCH
                     SYS_CALL       SWITCH
                     SEG_TRAP       SWITCH
                     NMI_INT        SWITCH
                     NVI_INT        SWITCH
                     VEC_FCW        WORD
                     VEC_PC         ARRAY [200  WORD]
```

```
                    1

                INTERNAL
                $SECTION DATA_DEC
                $ABS 0

0000               INTBUF    ARRAY [128 BYTE]
0080               OUTBUF    ARRAY [128 BYTE]
0100               RNGBUF    ARRAY [256 BYTE]
0200               MCZBUF    ARRAY [256 BYTE]
0300               BUFADR    WORD
0302               BUFSIZ    WORD
0304               INTPTR    WORD
0306               OUTPTR    WORD
0308               NXTPTR    WORD
030A               GETOUT    WORD
030C               MCZPUT    WORD
030F               MCZGET    WORD
0310               MFLAGS    WORD
0312               RETRY     WORD
0314               ADR_STR   WORD
0316               DWN_ADR   WORD
0318               START_ADR WORD

                EXTERNAL
                   LOAD_FILE  PROCEDURE

                $SECTION BOOTSTRAP_PROC
                $REL 0

                GLOBAL
0000            BOOTSTRAP  PROCEDURE
                !*****************************************
                 *                                       *
                 *  INIT_TEST: INITIALIZES MULTIPRO-     *
                 *         CESSOR SYSTEM WITH A COMMON   *
                 *         MEMORY MAP AND SYSTEM WIDE    *
                 *         KNOWLEDGE OF EACH UNIQUE      *
                 *         CPU ID; LOADS LOCAL AND       *
                 *         GLOBAL PORTIONS OF O/S;       *
                 *         AND TANSFERS CONTROL TO       *
                 *         LOCAL PORTION.                *
                 *                                       *
                 *****************************************!
                ENTRY
0000 34CC          LDAR      R12,BOOTSTRAP  ! SET NEW CODE_AREA !
0002 FFFC


                ! INITIALIZE SYSTEM CALL HANDLER IN PSA !
```

```
0004 7DA5      LDCTL    R10,PSAPOFF
0006 340B      LDAR     R11,SYSTEM_CALL
0008 0284
000A 76A1      LDA      R1,SYS_CALL(R10)
000C 000C
000E 7D15      LD       @R1,#%4000      ! LOAD FCW FOR SC !
0010 4000
0012 331B      LD       R1(#2),R11      ! LOAD PC FOR SC !
0014 0002

0016 340B      LDAR     R11,MEM_INT
0018 0382
001A 76A1      LDA      R1,VEC_PC(R10)
001C 001E
001E 331B      LD       R1(#14),R11     ! LOAD MEM VIOLATION !
0020 000E                                ! INT HANDLER IN PSA !

          ! INITIALIZE NORMAL STACK POINTER !
0022 210B      LD       R11,#%4500
0024 4500
0026 7DBF      LDCTL    NSP,R11

          ! MOVE CODE TO COMMON NORMAL/SYSTEM AREA !
0028 2101      LD       R1,#%4100       ! NEW CODE LOCATION   !
002A 4100
002C 34C4      LDAR     R4,NORM_SCRIBE  ! START OF CODE !
002E 001E
0030 3402      LDAR     R2,GSB          ! END OF CODE !
0032 0072

               DO
0034 214D         LD    R13,@R4
0036 2F1D         LD    @R1,R13
0038 6B42         CP    R2,R4           ! CK FOR END !
003A E603         JR    Z,NXT_SCB
003C A911         INC   R1,#2
003E A941         INC   R4,#2
0040 E8F9      OD                      ! CODE TRANSFERRED !

          NXT_SCB:
0042 2108      LD       R8,#%4100
0044 4100
0046 2103      LD       R3,#%411E      ! SET RTN ADR !
0048 411E
004A 1F80      CALL     @R8          ! SCRIBE MEMORY !
004C E82D      JR       TEST_LOCK
```

```
                !*****************************************
                *                                       *
                *  NORM_SCRIBE: SCRIBES MEMORY IN THE   *
                *      NORMAL MODE USING SYSTEM CALL    *
                *      INSTRUCTION FOR BUS LOCKS AND    *
                *      MODE CHANGES. THIS CODE SECTION  *
                *      WILL BE EXECUTED FROM A COMMON   *
                *      (NORMAL/SYSTEM) AREA OF MEMORY.  *
                *                                       *
                *****************************************!
               NORM_SCRIBE:
004E 2104          LD        R4,#%1000
0050 1000
0052 7D4A          LDCTL     FCW,R4          ! CHG TO NORMAL MODE !
0054 2101          LD        R1,#%AA55       ! NORMAL R/W PATTERN !
0056 AA55
0058 2104          LD        R4,#%F800       ! TOP BLOCK ADDR !
005A F800
005C 8D88          CLR       R9

               !******  CLEAR MEMORY NORMAL MODE  ******* !

                   DO
005E 6F41              LD     NORM_RW_PAT(R4),R1
0060 0004
0062 6F49              LD     NORM_CPU_CNT(R4),R9       ! CLEAR CPU_CNT !
0064 0006
0066 0304              SUB    R4,#%0800
0068 0800
006A F8F9          OD

               ! WAIT FOR OTHER CPU'S TO COMPLETE TASK !
               WAIT3:
006C 2103          LD         R3,#%1000
006F 1000
0070 7D3A          LDCTL      FCW,R3

0072 2103          LD         R3,#50
0074 0032
                   DO
0076 1904              MULT   RR4,#1
0078 0001
007A A330              DEC    R3
007C E601              JP     Z,SCBE
007E F8FB          OD

               !*******   SCRIBE  IN  NORMAL  MODE    *******!

               SCBE:
0080 2104          LD        R4,#%F800
```

```
0082 F800
                DO
0084 6148           LD      R8,NORM_RW_PAT(R4)
0086 0004
0088 9318           CP      R8,R1           ! CK FOR R/W !
008A F807           JR      NZ,NO_RW        ! NO R/W !

              ! CPU IDENTIFIES ACCESS TO THIS BLOCK !
008C 7F01           SC      #1              ! LOCK MULTIBUS !
008E 6148           LD      R8,NORM_CPU_CNT(R4)     ! GET CPU_NUM !
0090 0006
0092 1987           INC     R8,#1           ! INCREMENT !
0094 6F48           LD      NORM_CPU_CNT(R4),R8
0096 0006
0098 7F02           SC      #2              ! UNLOCK MULTIBUS !

                NO_RW:
009A 93D4           CP      R4,R13    ! CK FOR LAST BLK !
009C E603           JR      Z,TERMN         ! FINISHED !
009E 0304           SUB     R4,#%0800
00A0 0800
00A2 F8F0       OD

              TERMN:
00A4 7F03       SC          #3          ! CHG TO SYSTEM MODE !

              GSB:
00A6 9E08       RET

              !*****************************************
               *                                       *
               *  TEST_LOCK: ROUTINE TO GAIN ACCESS    *
               *       TO CONFIG_TABLE.                *
               *                                       *
               *  REG USE: INPUT  R7 = LOW GLOBAL MEM  *
               *                                       *
               *****************************************!
              TEST_LOCK:
00A8 7678       LDA         R8,TABLE_LOCK(R7) ! LOCK ADR !
00AA 0008

              ! MAIN LOCK TESTING LOOP !
                DO
00AC DF8F           CALR    TEST_N_SET    ! ACCESS CONFIG_TABLE !
00AE FD03           JR      PL,TBL_ACCESS ! GOT EXCLU. ACCESS !

              ! DELAY BEFORE NEXT ATTEMPT !
00B0 1904           MULT        RR4,#1
00B2 0001
00B4 F8FB       OD
```

```
                !*****************************************
                *                                       *
                *  TBL_ACCESS: ROUTINE TO DETERMINE     *
                *     DIRECTOR_CPU AND MEMBER_CPU'S.    *
                *                                       *
                *****************************************!
                TBL_ACCESS:
00B6 7673       LDA     R3,CPU_CNT(R7)
00B8 000A
00BA 213E       LD      R14,@R3
00BC 8B4E       CP      R14,R4          ! CK IF OWN CPU NO. !
00BE F604       JR      Z,OWN_ID
00C0 4D75       LD      TABLE_LOCK(R7),#0
00C2 00C8
00C4 0000
00C6 E8F0       JR      TEST_LOCK

                OWN_ID:
00C8 0B0E       CP      R14,#0          ! CHECK IF LOG_CPU 0? !
00CA 0000

00CC F601       JR      Z,BOOTLOAD_CPU  ! YES !
00CE E811       JR      MEMBER_CPU      ! NO  !

                ! *****    BOOTLOAD_CPU  START    *****  !

                !*****************************************
                *                                       *
                *  BOOTLOAD_CPU: CPU ASSUMES ROLE AS    *
                *       INITIALIZATION COORDINATOR.     *
                *       CPU DETERMINES NUMBER OF CPU    *
                *       IN SYSTEM AND INITIALIZES THE   *
                *       CONFIG_TABLE ACCORDINGLY. IT    *
                *       THEN IDENTIFIES ITSELF AND      *
                *       WAITS FOR MEMBER_CPU'S TO       *
                *       IDENTIFY THEMSELVES; AND THEN   *
                *       CONSOLIDATES CONFIG_TABLE DATA  *
                *       INTO SYSTEM TABLE (MIN_CONFIG_  *
                *       TBL).                           *
                *                                       *
                *  REG USE: INPUT  R5 = NUMBER CPU'S    *
                *                  R6 = LOW LOCAL MEM   *
                *                  R7 = LOW GLOBAL MEM  *
                *                  R4 = LOG_CPU NO.     *
                *                                       *
                *****************************************!
                BOOTLOAD_CPU:
                ! LOAD CPU_CNT WITH NEXT LOG_CPU NUMBER !
00D0 4D75       LD      CPU_CNT(R7),#1    ! LOAD LOG_CPU 1 !
```

```
00D2 000A
00D4 0001

                ! COMPLETE OWN CONFIG_TABLE ENTRY !
00D6 7673       LDA     R3,CPU_LIST(R7)     ! LOG_CPU 0 !
00D8 00DC
00DA 763E       LDA     R14,MEM_MAP(R3)
00DC 00CA
00DE DF62       CALL    MAP_MEMORY    ! ENTER MEM MAP !

                ! UNLOCK CONFIG_TABLE !
00E0 7678       LDA     R8,TABLE_LOCK(R7) ! LOCK ADR !
00E2 0008
00E4 0D85       LD      @R8,#0          ! CLEAR LOCK !
00E6 0000

                ! WAIT FOR ALL CPU'S TO IDENTIFY THEMSELVES !
00E8 7678       LDA     R8,CPU_CNT(R7)  ! CPU_CNT ADR !
00EA 00CA
00EC A159       LD      R9,R5       ! GET TOTAL PHYS_CPU !
00EE DF02       CALR    WAIT_COMP    ! WAIT FOR ALL !
00F0 E83A       JR      COMBIN

                ! *****     MEMBER_CPU  START    ***** !
                !*****************************************
                 *                                       *
                 *  MEMBER_CPU: IDENTIFIES ITSELF IN     *
                 *       CONFIG_TABLE AND ENTERS MEM_    *
                 *       MAP. WAITS FOR DIRECTOR_CPU     *
                 *       TO SIGNAL TO CONTINUE.          *
                 *                                       *
                 *  REG USE: INPUT  R4 = LOG_CPU NO.     *
                 *                                       *
                 *****************************************!
                MEMBER_CPU:
00F2 A14E       LD      R14,R4        ! SAVE LOG_CPU NO. !
00F4 767A       LDA     R10,CPU_LIST(R7)  ! BASE OF ENTRY !
00F6 00CC
00F8 8D28       CLR     R2
                DO
00FA 0102           ADD     R2,#74        ! DETERMINE BASE OF !
00FC 004A
00FE ABE0           DEC     R14            ! ENTRY !
0100 E601           JR      Z,CONT
0102 E8FB       OD

                CONT:
0104 812A       ADD     R10,R2        ! COMPUTE BASE ADR !
0106 A1A3       LD      R3,R10
```

```
                ! ENTER MEM_MAP IN TABLE ENTRY !
0108 76AE       LDA     R14,MEM_MAP(R10)  ! MEM_MAP BASE !
010A 00CA
010C DF79       CALR    MAP_MEMORY

                ! LOAD CPU_CNT WITH NEXT LOG_CPU NUMBER !
010E 7679       LDA     R8,CPU_CNT(R7)
0110 00CA
0112 2189       LD      R9,@R8          ! CPU COUNT !
0114 A990       INC     R9              ! ADD ONE, !
0116 2F89       LD      @R8,R9          ! AND PUT BACK !

                ! UNLOCK CONF_TABLE FOR OTHER PROCESSORS !
0118 7678       LDA     R8,TABLE_LOCK(R7)
011A 00C8
011C 0D85       LD      @R8,#0          ! CLEAR LOCK !
011E 0000

                ! WAIT FOR SIGNAL TO PROCEED WITH BOOTLOAD !
0120 76A8       LDA     R8,SIGNAL(R10)  ! OWN SIGNAL ADR !
0122 0000
                DO
0124 2182           LD      R2,@R8      ! TEST SIGNAL !
0126 0B02           CP      R2,#1
0128 0001
012A E6C1           JR      Z,DWN_LD
012C E8FB       OD

                DWN_LD:
                ! DOWN-LOAD PROC_LOCAL INTO LOCAL MEMORY !
012E 76A1       LDA     R1,MSG_BLK(R10)   ! OWN MSG_BLK !
0130 0004
0132 2112       LD      R2,@R1          ! CURRENT CODE ADR !
0134 3113       LD      R3,R1(#2)       ! LOCAL LOAD ADR !
0136 0002
0138 311E       LD      R14,R1(#4)      ! NUMBER OF BYTES !
013A 0004

013C BA21       LDIRB   @R3,@R2,R14     ! MOVE CODE !
013E 0F30

                ! CLEAR SIGNAL !
0140 0D85       LD      @R8,#0
0142 0000

                ! SIGNAL DIRECTOR_CPU THAT THIS CPU DONE !
0144 DED9       CALR    TEST_N_SET      ! ACCESS TO !
                                        ! CONFIG_TABLE !
0146 7673       LDA     R14,CPU_CNT(R7)
0148 000A
```

```
214A 2159      LD      R9,@R14         ! LOG_CPU NUMBER !
214C A990      INC     R9
214E 2FE9      LD      @R14,R9         ! INCREMENT !

0150 767E      LDA     R14,TABLE_LOCK(R7)
0152 0008
0154 0DF5      LD      @R14,#0         ! CLEAR LOCK !
0156 0000

               ! WAIT FOR SIGNAL TO TRANSFER CONTROL !
                  DO
0158 2182            LD    R2,@R8      ! TEST SIGNAL !
015A 0B02            CP    R2,#1
015C 0001
015E F601            JR    Z,TRN_CNTL
0160 F8FB         OD

               TRN_CNTL:
               ! TRANSFER CONTROL TO O/S !
0162 2112         LD       R2,@R1      ! OWN MSG_BLK !
0164 1E28         JP       @R2         ! START OF O/S !

               ! *****   END OF MEMBER_CPU   ***** !

               !****************************************
                *                                      *
                *     GLOBAL INITIALIZATION STAGE      *
                *                                      *
                ****************************************!

               COMBIN:

               !****************************************
                *  CONSOLIDATE CPU_ENTRY'S INTO TABLE  *
                ****************************************!

               ! CONSOLIDATE LOG_CPU TO PHYS_ID IN TABLE !
0166 A171         LD       R1,R7
0168 7671         LDA      R1,CPU_LIST(R7)  ! TABLE ENTRY BASE !
016A 000C
016C A159         LD       R9,R5
                  DO
016E 0101            ADD   R1,#74       ! ADR OF NEXT ENTRY !
0170 004A
0172 AB90            DEC   R9
0174 F601            JR    Z,HAVE_ADR
0176 F8FB         OD
               HAVE_ADR:
                                       ! ADDRESS !
0178 7679         LDA      R9,CPU_LIST(R7) ! GET START OF !
```

```
017A 000C
                                        ! CPU_ENTRY'S !
017C 7692       LDA     R2,CPU_ID(R9)
017E 0002
0180 A154       LD      R4,#5

                DO
0182 212D           LD      R13,@R2     ! STORE PHYS_ID !
0184 2F1D           LD      @R1,R13
0186 A911           INC     R1,#2
0188 0172           ADD     R2,#74
018A 004A
018C AB40           DEC     R4
018E 0B04           CP      R4,#0       ! CK IF FINISHED !
0190 0000
0192 E601           JR      Z,MEM_CONSOL
0194 E8F6       OD

            ! CONSOLIDATE MEM_MAP'S INTO MIN_CONFIG_TBL !
            MEM_CONSOL:
0196 2104       LD      R4,#32
0198 0020
019A 7679       LDA     R9,CPU_LIST(R7)
019C 000C
019E 7692       LDA     R2,MEM_MAP(R9)  ! LOG_CPU 0 !
01A0 000A

                DO
01A2 212D           LD      R13,@R2     ! COPY LOG_CPU 0 VALUES !
01A4 2F1D           LD      @R1,R13
01A6 A911           INC     R1,#2
01A8 A921           INC     R2,#2
01AA AB40           DEC     R4
01AC E601           JR      Z,NXT_CPU   ! GET NEXT LOG_CPU !
01AE E8F9       OD

            NXT_CPU:
            ! MAIN LOOP FOR CONSOLIDATION OF REMAINING MAPS !
01B0 A15B       LD      R11,R5
01B2 ABB0       DEC     R11,#1
01B4 E619       JR      Z,LL            ! NO MORE MAPS TO DO !

                DO
01B6 210D           LD      R13,#32
01B8 0020
01BA A129           LD      R9,R2
01BC 7692           LDA     R2,MEM_MAP(R9)  ! NEXT LOG_CPU !
01BE 000A
01C0 0301           SUB     R1,#64      ! ADR NEW TBL MEM_MAP !
01C2 0040
```

```
                    DO
01C4 2123               LD      R3,@R2    ! NEW VALUES !
01C6 2114               LD      R4,@R1    ! ORIG VALUE !
01C8 9A34               CPB     RH4,RH3   ! COMPARE !
01CA EF01               JR      NC,COMP_LOW
01CC A034               LDB     RH4,RH3   ! SAVE NEW VALUE !
                    COMP_LOW:
01CE 8ABC               CPB     RL4,RL3
01D0 EF01               JR      NC,AGAIN
01D2 A0BC               LDB     RL4,RL3   ! SAVE NEW VALUE !
                    AGAIN:
01D4 2F14               LD      @R1,R4    ! SAVE MAP BLOCK !
01D6 A911               INC     R1,#2
01D8 A921               INC     R2,#2
01DA ABD0               DEC     R13
01DC F601               JR      Z,OVR     ! GET NEXT MEM_MAP!
01DE F8F2           OD

                  OVR:
01E0 ABB0             DEC     R11,#1
01E2 F601             JR      Z,LL    ! FINISHED !
01E4 F8E8         OD

                !*****************************************
                 *                                       *
                 *    OPERATING SYSTEM CORE IMAGE        *
                 *              LOAD  STAGE              *
                 *                                       *
                 *****************************************!



                LL:
                ! SUPERVISE PROC_LOCAL LOADING !
01E6 3404         LDAR      R4,PROC_LOCAL  ! LOAD KERNEL !
01E8 01BE
01EA DF6F         CALR      COORD_DWN_LD

                ! SAVE ENTRY ADDRESS TO KERNEL !
01EC 61E4         LD        R4,DWN_ADR(R14)
01EE 0316
01F0 6FE4         LD        START_ADR(R14),R4
01F2 0318


                ! LOAD PROC_GLOBAL OF O/S !
01F4 3404         LDAR      R4,PROC_GLOBAL  ! LOAD SUPERVISOR !
01F6 01BF
01F8 DF76         CALR      COORD_DWN_LD
```

```
                ! SIGNAL ALL CPU TO TRANSFER CONTROL TO O/S !
01FA 6100       LD      R0,START_ADR
01FC 0318
01FE 6174       LD      R4,CPU_LIST(R7)  ! LIST BASE ADR !
0200 000C
0202 A153       LD      R3,R5            ! NO. CPU'S !
                DO
0204 6F40         LD    MSG_BLK(R4),R0   ! STORE START ADR !
0206 0004
0208 AB30         DEC   R3
020A E603         JR    Z,END_SIG
020C 0104         ADD   R4,#74           ! NEXT CPU ENTRY !
020E 004A
0210 F8F9       OD
                END_SIG:
0212 DF52       CALR    SIGNAL_CPU
0214 2104       LD      R4,#0          ! SAVE LOG_CPU NO. !
0216 0000

                ! TRANSFER CONTROL TO START OF CODE !
0218 A102       LD      R2,R0
021A 1E28       JP      @R2

                ! *****    END  DIRECTOR_CPU    ***** !

021C            END BOOTSTRAP

021C            MAP_MEMORY  PROCEDURE
                !*************************************
                 *                                   *
                 *  MAP_MEMORY: MAPS CPU MEMORY ACCESS  *
                 *        BY DOMAIN, AS TO LOCAL (1),   *
                 *        GLOBAL (2), DUAL_USE (3),     *
                 *        NON_USE (4), NON_ACCESS (5).  *
                 *                                   *
                 *  REG USE: INPUT  R14 = ADR MEM_MAP   *
                 *                                   *
                 *************************************!
                ENTRY
                ! MOVE CODE TO LOCAL, NORMAL MEMORY AREA !
021C 2101       LD      R1,#%4100      ! NEW LOCATION !
021E 4100
0220 3404       LDAR    R4,NORM_MAP    ! START OF CODE !
0222 0034
0224 3402       LDAR    R2,NORM_RTN    ! END OF CODE !
0226 0084

                DO
0228 2149         LD    R9,@R4         ! MOVE CODE !
```

```
022A 2F19          LD      @R1,R9
022C 9B42          CP      R2,R4          ! CK FOR END !
022E F603          JR      Z,BEGIN_MAP
0230 A911          INC     R1,#2
0232 A941          INC     R4,#2
0234 F8F9      OD

               ! BEGIN MAPPING NORMAL MODE MEM ACCESS !
               BEGIN_MAP:
0236 A1D4      LD          R4,R13    ! LOW BLK ACCESS !
0238 A1F1      LD          R1,R14
               DO
023A 8D44        TEST      R4
023C E604        JR        Z,FND_BLK
023E 0304        SUB       R4,#%0800
0240 0800
0242 A911        INC       R1,#2
0244 F8FA      OD
               FND_BLK:
0246 A11E      LD          R14,R1
0248 A1D4      LD          R4,R13
               ! R1 = MAP END BLK,  R4 = ADR !
024A 2109      LD          R9,#%F804
024C F804
024E A943      INC         R4,#4

               ! MAP AND ENTER FROM NORMAL MODE !
0250 2108      LD           R8,#%4100
0252 4100
0254 1F80      CALL        @R8
0256 9E08      RET

0258           END MAP_MEMORY

0258           NORM_MAP PROCEDURE
               !*****************************************
                *                                       *
                *  NORM_MAP: MAPS MEMORY ACCESS IN THE  *
                *       NORMAL MODE INTO CONFIG_TABLE   *
                *       FOR CPU.                        *
                *                                       *
                * REG USE: INPUT  R1 = MEM_MAP ADR      *
                *                 R4 = MEM ADR          *
                *                 R9 = STOP ADR         *
                *                                       *
                *****************************************!
               ENTRY
0258 2102      LD          R2,#%AA55     ! NEW R/W PATTERN !
025A AA55
025C 2110      LD          R0,@R1
```

```
                    DO
025E 210A               LD      R10,#%1700
0260 1000
0262 7D8A               LDCTL   FCW,R10         ! CHG TO NORMAL MODE !

0264 2143               LD      R3,@R4
0266 8B23               CP      R3,R2           ! TEST FOR R/W !
0268 EE17               JR      NZ,NC_ACCESS    ! NO R/W/ !
026A 4143               LD      R3,R4
026C A931               INC     R3,#2           ! CK FOR LOCAL !
026E 213B               LD      R11,@R3
0270 0B0B               CP      R11,#1          ! CPU_NUM = 1? !
0272 0001
0274 EF0B               JR      NZ,GLOB_ACC

                    ! LOCAL ACCESS !
0276 4143               LD      R3,R4
0278 0303               SUB     R3,#4           ! CK FOR SYS MODE R/W !
027A 0004
027C 213B               LD      R11,@R3         ! GET PATTERN !
027E 0B01               CP      R1,#%55AA
0280 55AA
0282 E602               JR      Z,DUAL_MODE
0284 C901               LDB     RL0,#%01        ! ENTER LOCAL ID !
0286 E809               JR      CYCLE

                    DUAL_MODE:
0288 C803               LDB     RL0,#%03        ! ACCESS BY N/S MODES !
028A E807               JR      CYCLE

                    GLOB_ACC:
028C 8B5B               CP      R11,R5          ! CK FOR GLOBAL !
028E EF02               JR      NZ,NON_USE
0290 C802               LDB     RL0,#%02        ! ENTER GLOBAL ID !
0292 E803               JR      CYCLE

                    NON_USE:
0294 C804               LDB     RL0,#%04        ! ENTER NON_USE ID !
0296 E801               JR      CYCLE

                    NO_ACCESS:
0298 C805               LDB     RL0,#%05        ! ENTER NON_ACCESS ID !

                    CYCLE:
029A 7F03               SC      #3              ! CHG TO SYSTEM MODE !
029C 2F10               LD      @R1,R0
029E 8B94               CP      R4,R9           ! CK FOR COMPLETION !
02A0 E605               JR      Z,NORM_RTN
02A2 0104               ADD     R4,#%0800       ! NEXT MEM BLOCK !
```

```
02A4 0000
02A6 A911          INC    R1,#2
02A8 2110          LD     R0,@R1          ! NEXT MAP BLOCK !

02AA E8D9       OD

                NORM_RTN:
02AC 9E08       RET

02AE            END NORM_MAP

02AE            SYSTEM_CALL  PROCEDURE
                !*****************************************
                 *                                       *
                 *  SYSTEM_CALL: AFFECTS PRIVILEDGED     *
                 *      INSTRUCTION EXECUTION IN THE     *
                 *      NORMAL MODE.                     *
                 *                    1 = BUS LOCK       *
                 *                    2 = BUS UNLOCK     *
                 *                    3 = CHG FCW        *
                 *                                       *
                 *****************************************!
                ENTRY
02AE 93F0          PUSH     @R15,R0        ! SAVE WORK REG !
02B0 31F0          LD       R0,R15(#2)       ! GET INSTRUCTION !
02B2 0002
02B4 8C08          CLRB     RH0            ! GET EXECUT. CODE !
02B6 0B00          CP       R0,#1          ! CK FOR BUS LOCK !
02B8 0001
02BA EE04          JR       NZ,CK2
02BC 3B26          OUT      BUS_LOCK,R2    ! LOCK MULTIBUS !
02BE FFF9
02C0 97F0          POP      R0,@R15        ! RESTORE WORK REG !
02C2 7B00          IRET

                CK2:
02C4 0B00          CP       R0,#2           ! CK FOR UNLOCK BUS !
02C6 0002
02C8 EE04          JR       NZ,CK3
02CA 3B26          OUT      BUS_UNLOCK,R2 ! UNLOCK MULTIBUS !
02CC FFF8
02CE 97F0          POP      R0,@R15        ! RESTORE WORK REG !
02D0 7B00          IRET

                CK3:
02D2 0B00          CP       R0,#3           ! CK FOR MODE CHG !
02D4 0003
02D6 EE08          JR       NZ,NONE
02D8 93FB          PUSH     @R15,R11
02DA 210B          LD       R11,#%5000
```

```
02DC 5000
02DE 33FB       LD      R15(#6),R11
02E0 0006
02E2 97FB       POP     R11,@R15
02E4 97F0       POP     R0,@R15         ! RESTORE WORK REG !
02E6 7B00       IRET

                NONE:
02E8 97FC       POP     R0,@R15         ! RESTORE WORK REG !
02EA 7B00       IRET

02EC            END SYSTEM_CALL

02EC            WAIT_COMP PROCEDURE
                !*****************************************
                *                                       *
                *  WAIT_COMP: WAIT FOR SPECIFIED MEM    *
                *      LOCATION (R8) TO CONTAIN THE     *
                *      SPECIFIED VALUE (R9) BEFORE      *
                *      RETURN.                          *
                *                                       *
                *  REG USE: INPUT  R8 = ADDR            *
                *                  R9 = VALUE           *
                *                                       *
                *****************************************!
                ENTRY
                   DO
02EC 218D             LD    R13,@R8     ! GET VALUE !
02EE 839D             CP    R13,R9      ! CK FOR MATCH !
02F0 F603             JR    Z,GOT_SIG
02F2 1904             MULT  RR4,#1      ! DELAY !
02F4 0001
02F6 F8FA          OD

                GOT_SIG:
02F8 0D85       LD      @R8,#0          ! CLR MSG BLK !
02FA 0000
02FC 9E08       RET

02FE            END WAIT_COMP
```

```
                GLOBAL
02FF            CPU_WAIT  PROCEDURE
                !*****************************************
                 *                                       *
                 *  CPU WAITS APPROX. 2MSEC FOR ALL      *
                 *  CPU'S TO COMPLETE THE SAME TASK      *
                 *  BEFORE CONTINUING.                   *
                 *                                       *
                 *****************************************!
                ENTRY
02FF 2103          LD        R3,#50
0300 0032
                   DO
0302 1904             MULT   RR4,#1
0304 0001
0306 AB30             DEC    R3
0308 E601             JR     Z,RETRN
030A F8FB          OD
                RETRN:
030C 9E08          RET
030E            END CPU_WAIT

030E            COORD_DWN_LD  PROCEDURE
                !*****************************************
                 *                                       *
                 *  LOAD_LOCAL: LOADS PROCESSOR LOCAL    *
                 *       PORTION OF O/S INTO GLOBAL MEM  *
                 *       AND SUPERVISES LOADING BY ALL   *
                 *       CPU'S. SETS DOWN-LOAD INST. IN  *
                 *       MSG_BLK OF EACH CPU.            *
                 *                                       *
                 *****************************************!
                ENTRY
030E A173          LD        R3,R7
0310 0103          ADD       R3,#%0800     ! GLOBAL LOAD ADR !
0312 0800

0314 8D88          CLR       R8            ! RESET BYTE COUNT !

                ! RESTORE PREVIOUS DATA AREA !
0316 7D15          LDCTL     R1,PSAPOFF
0318 611E          LD        R14,DATA_AREA(R1)
031A 0000

                ! LOAD FILE FROM MCZ !
031C 5FC0          CALL      LOAD_FILE(R12)
031F 0000*

                ! SET DOWN-LOAD INSTRUCTIONS FOR OTHER CPU !
0320 A173          LD        R3,R7
```

```
0322 0103       ADD         R3,#%0800     ! GLOBAL LOAD ADR !
0324 0800
0326 7671       LDA         R1,CPU_LIST(R7)      ! BASE OF !
0328 000C
                                          ! TABLE ENTRIES !
032A 7612       LDA         R2,MSG_BLK(R1) ! MSG_BLK LOG_CPU 2 !
032C 0004
032E 61E4       LD          R4,DWN_ADR(R14)  ! PROC_LOAD DOWN- !
0330 0316
                                        ! LOAD ADDRESS !
                                        ! R8 = SIZE !
0332 A156       LD          R6,R5         ! TOTAL NO. CPU !

                DO
0334 2F23           LD      @R2,R3    ! LOAD GLOBAL ADR MSG1 !
0336 3324           LD      R2(#2),R4 !  LOCAL ADR MSG2 !
0338 0002
033A 3328           LD      R2(#4),R8  ! SIZE OF CODE MSG3 !
033C 0004
033E AB60           DEC     R6
0340 E603           JR      Z,DO_SELF
0342 0102           ADD     R2,#74    ! ADR NEXT LOG_CPU MSG_BLK !
0344 004A
0346 F8F6       CD

            DO_SELF:
            ! LOAD PROCEDURE LOCAL CODE INTO OWN LOCAL MEM !
0348 7671       LDA         R1,CPU_LIST(R7)
034A 000C
034C 7612       LDA         R2,MSG_BLK(R1)  ! OWN MSG_BLK ADR !
034E 0004
0350 2121       LD          R1,@R2        ! SOURCE ADDRESS !
0352 3123       LD          R3,R2(#2)     ! DESTINATION ADR !
0354 0002
0356 3124       LD          R4,R2(#4)     ! NUMBER BYTES !
0358 0004

035A BA11       LDIRB       @R3,@R1,R4    ! DOWN-LOAD CODE !
035C 0430

            ! SET CONFIG_TABLE CPU_CNT TO LOG_CPU 1 !
035E 4D75       LD          CPU_CNT(R7),#1
0360 0004
0362 0001

            ! SIGNAL ALL CPU TO PROCEED WITH DOWN-LOAD !
0364 DFFB       CALR        SIGNAL_CPU

            ! WAIT FOR ALL CPU TO COMPLETE !
0366 7678       LDA         R8,CPU_CNT(R7)
```

```
0368 0001
036A A159       LD      R9,R5           ! NUMBER CPU !
036C D041       CALR    WAIT_COMP  ! WAIT FOR COMPLETION !

036E 9E08       RET     ! RETURN TO DIRECTOR_CPU !

0370        END COORD_DWN_LD

0372        SIGNAL_CPU  PROCEDURE
            !*****************************************
             *                                       *
             *  SIGNAL_CPU: PLACES SIGNAL (#1) IN    *
             *       SIGNAL BLOCK FOR EACH CPU EN-   *
             *       TRY IN CONFIG_TABLE.            *
             *                                       *
             *****************************************!
            ENTRY
            ! SIGNAL ALL CPU TO DOWN-LOAD !
0370 7673       LDA     R3,CPU_LIST(R7)
0372 000C
0374 0103       ADD     R3,#74          ! LOG_CPU 1 ENTRY !
0376 004A
0378 A154       LD      R4,R5           ! TOTAL NO. CPU !
037A AB40       DEC     R4
037C E608       JR      Z,ALL_SIG

                DO
037E 4D35           LD      SIGNAL(R3),#1  ! LOAD SIGNAL !
0380 0000
0382 0001
0384 AB40           DEC     R4
0386 E603           JR      Z,ALL_SIG
0388 0103           ADD     R3,#74      ! NEXT LOG_CPU ADR !
038A 004A
038C F8F8       OD

            ALL_SIG:
038E 9E08       RET

0390        END SIGNAL_CPU
```

```
0390            TEST_N_SET  PROCEDURE
                !*****************************************
                 *                                       *
                 * TEST_N_SET: ROUTINE UTILIZING HDWR    *
                 *             TSET INSTRUCTION AND      *
                 *             BUS LOCKING ABILITY TO    *
                 *             AFFECT MUTUAL EXCLUSION   *
                 *             ON ACCESS TO COMMON       *
                 *             DATA STRUCTURES LOCK.     *
                 *                                       *
                 *  REG USE: INPUT  R8 = ADR OF LOCK     *
                 *                                       *
                 *****************************************!
                ENTRY
0390 3B26          OUT       BUS_LOCK,R2   ! LOCK SYSTEM BUS !
0392 FFF9
0394 0D86          TSET      @R8           ! TEST TABLE LOCK !
0396 3B26          OUT       BUS_UNLOCK,R2 ! UNLOCK SYSTEM BUS !
0398 FFF8
039A 9E08          RET
039C            END TEST_N_SET

                GLOBAL
039C            MEM_INT  PROCEDURE
                !*****************************************
                 *                                       *
                 *  MEM_INT: INTERRUPT HANDLER FOR AN    *
                 *           ILLEGAL LOCAL MEMORY        *
                 *           ACCESS IN THE NORMAL MODE   *
                 *           SAVES MEMORY ADDRESS OF     *
                 *           NEXT BLOCK.                 *
                 *                                       *
                 *  REG USE: INPUT  R3 = RTN POINT       *
                 *                  R4 = FAULT ADR       *
                 *                                       *
                 *****************************************!
                ENTRY
039C 01C4          ADD       R4,#%0800     ! RESTORE BLOCK ADR !
039E 0800
03A0 A14D          LD        R13,R4        ! SAVE ADR !
03A2 010F          ADD       R15,#6        ! RESTORE STACK !
03A4 0006
03A6 1E38          JP        @R3           ! RTN TO SYS CODE !
03A8            END MEM_INT

                PROC_LOCAL:
03A8 0D            BVAL    %0D
03A9 4C4F          WVAL    'LO'
03AC 4144          WVAL    'AD'
03AE 204B          WVAL    ' K'
```

```
03B0 4552       WVAL    'ER'
03B2 4E45       WVAL    'NE'
03B4 4C20       WVAL    'L '
03B6 20         BVAL    ' '

                PROC GLOBAL:
03B7 11         BVAL    %11
03B8 4C4F       WVAL    'LO'
03BA 4144       WVAL    'AD'
03BC 2053       WVAL    ' S'
03BE 5550       WVAL    'UP'
03C0 4552       WVAL    'ER'
03C2 5649       WVAL    'VI'
03C4 534F       WVAL    'SO'
03C6 5220       WVAL    'R '
03C8 20         BVAL    ' '

                END BOOTSTRP
```

2. SUPPORT1 MODULE (same as Appendix C)

3. SUPPORT2 MODULE (same as Appendix C)

4. SUPPORT3A MODULE

```
Z8000'SM  2.02
LOC    OBJ CODE     STMT SOURCE STATEMENT

                      1 SUPPORT3A MODULE
                  $LISTON  $TTY

                  !****************************************
                   *                                      *
                   *  SUPPORT3A MODULE: MODULE THREE FOR  *
                   *       SECONDARY STORAGE PRIMITIVES   *
                   *       FUNCTIONS SUPPORT. STRICTLY    *
                   *       HARDWARE DEPENDENT; SHOULD     *
                   *       MEET STORAGE DEVICE REQUIRE-   *
                   *       MENTS FOR INTERFACING.         *
                   *                                      *
                   *  NOTE: DUPLICATE OF MONITOR LOAD_CMD *
                   *        MODULE.                       *
                   *                                      *
                   ****************************************!


                  CONSTANT

                     RXR       :=  2
                     TXR       :=  0
                     PAR       :=  7
                     PORTAD    :=  %FFD9
                     PORTBD    :=  %FFE1
                     PORTAC    :=  %FFDB
                     PORTBC    :=  %FFE3

                     IDPORT    :=  %FFCB
                     ICPORT    :=  %FFC9

                     TCMD    :=  %FFD2
                     TDTA    :=  %FFD0

                     BUS_LOCK :=%FFF9
                     BUS_UNLOCK :=%FFF8
                     VINTR     :=  %(2)0001000000000000
                     VIBIT     :=  12
```

```
                ESCAPE    :=  %1B
                BS        :=  %08
                LINDEL    :=  %7F
                CR        :=  %0D
                LF        :=  %0A
                TXOFCH    :=  %13
                TXONCH    :=  %11
                INSIZ     :=  128         ! INTBUF SIZE !
                OUTSIZ    :=  128         ! OUTBUF SIZE !
                RBSIZ     :=  256         ! RING BUFFER SIZE !
              ! BIT POSITIONS IN MONITOR FLAG WORD !
                TRPMDE    :=  0
                ISTOP     :=  1
                OSTOP     :=  2
                SNDMDE    :=  3
                LDMDE     :=  4
                ESC       :=  5
                TXMSK     :=  %6

              EXTERNAL    PRNTBF          PROCEDURE
              EXTERNAL    GETNXT          PROCEDURE
              EXTERNAL     ERROR           LABEL
              EXTERNAL     SNDCHR          PROCEDURE
              EXTERNAL     GETADR          PROCEDURE
              EXTERNAL     GMCZAD          PROCEDURE
              EXTERNAL     SNDMCZ          PROCEDURE
              EXTERNAL     CONVERT         PROCEDURE
              EXTERNAL     PBUFNC          LABEL
              EXTERNAL    SNDMSG          PROCEDURE
              EXTERNAL    CONVW           PROCEDURE

              INTERNAL
              $SECTION DATA_DEC
              $ABS 0

0000            INTBUF    ARRAY [128 BYTE]
0080            OUTBUF    ARRAY [128 BYTE]
0100            RNGBUF    ARRAY [256 BYTE]
0200            MCZBUF    ARRAY [256 BYTE]
0300            BUFADR    WORD
0302            BUFSIZ    WORD
0304            INTPTR    WORD
0306            OUTPTR    WORD
0308            NXTPTR    WORD
030A            GETOUT    WORD
030C            MCZPUT    WORD
030E            MCZGET    WORD
0310            MFLAGS    WORD
0312            RETRY     WORD
0314            ADR_STR   WORD
```

```
0316            DWN_ADR  WORD
0318            START_ADR WORD

                $SECTION LOAD_PROC
                $REL 0


                GLOBAL
0000            FNAME PROCEDURE
                !*****************************************
                 *                                       *
                 *   FNAME: RESETS TWO PTRS TO MCZBUF    *
                 *          AND CHECKS FOR FILENAME.     *
                 *                                       *
                 *****************************************!
                ENTRY
0000 4DE8          CLR       MCZGET(R14)
0002 030E
0004 4DF8          CLR       MCZPUT(R14)          ! RESET BUFFER !
0006 030C
0008 34CA          LDA       R10,R12(#GETNXT)
000A 0000*
000C 1FA0          CALL      @R10                 ! SKIP CMD ARG !
000E 0A08          CPB       RL0,#'A'
0010 4141
0012 E711          JR        C,DUN
0014 0A08          CPB       RL0,#'Z'+1
0016 5B5B
0018 EF0E          JR        NC,DUN               ! 1ST CHR IN (A..Z) !
001A 76CA          LDA       R10,GETNXT(R12)
001C 0000*
001E 1FA0          CALL      @R10                 ! SKIP TO NEXT ARG !
0020 E607          JR        Z,NO_ADR             ! NO NEXT ARG !
0022 76CA          LDA       R10,GETADR(R12)
0024 0000*
0026 1FA0          CALL      @R10                 ! GET USER SPECIFIED !
                                                  ! ADDRESS !
0028 A13B          LD        R11,R3               ! SAVE USER ADR !
002A 6FE3          LD        ADR_STR(R14),R3
002C 0314
002E 9E08          RET

                NO_ADR:
0030 210B          LD        R11,#%FFFE           ! SIGNAL TO USE MCZ !
0032 FFFE                                         ! ADDRESS !
0034 9E08          RET

                DUN:
0036 3D98          CLR       R9
```

```
0038 34CA     LDA       R10,R12(#ERROR)
003A 0000*
003C 1EA8     JP        @R10          ! ERROR, RTN TO EXEC !
003E          END FNAME

              GLOBAL
003F          CMDPAS PROCEDURE
              !*****************************************
               *                                       *
               *  CMDPAS: LOAD CMD PASSING MECHANISM   *
               *          SENDS 'B;' PLUS CONS CMD     *
               *  LINE TO MCZ AND CKS RESPONSES FOR    *
               *  GOOD Z80 PROGRAM LOAD.               *
               *                                       *
               *  REG USE:  RETURN   NZ IF Z80 LOADED  *
               *                     Z  IF NOT         *
               *                                       *
               *****************************************!
              ENTRY
0040 67E5     BIT       MFLAGS(R14),#ESC  ! CK FOR ESCAPE !
0042 0310
0044 E602     JR        Z,GCMD
0046 8D41     SETFLG    Z
0048 9E08     RET

              GCMD:
004A C242     LDB       RH2,#'B'
004C CA3B     LDB       RL2,#';'
004E 6FE2     LD        OUTBUF(R14),R2     ! LOAD INIT 'B;' !
0050 0080
                                           ! FOR BRIEF MODE !
0052 76F2     LDA       R2,OUTBUF(R14)
0054 0080
0056 A921     INC       R2,#2
0058 76E1     LDA       R1,INTBUF(R14)
005A 0000
005C 2100     LD        R0,#%40        ! LD CMD IN OUTBUF !
005E 0040
0060 BB11     LDIR      @R2,@R1,R0
0062 0020

0064 76F1     LDA       R1,OUTBUF(R14)
0066 0087
0068 0101     ADD       R1,#%80
006A 0080
006C 6FE1     LD        OUTPTR(R14),R1
006E 0306

0070 DFB6     CALR      OUTSTM         ! OUTPUT BUFFER !
0072 DFE7     CALR      SKIPLN         ! SKIP MCZ ECHO !
```

```
0072 DFD4      CALR    MCZCOM          ! WAIT RESPONSE !
0074 0A09      CPB     RL1,#'B'
0076 4242
0078 EE02      JR      NZ,LDSTAT
007A DFEC      CALR    SKIPLN          ! SKIP MCZ ECHO !
007C DFD9      CALR    MCZCOM          ! WAIT RESPONSE !


               ! VERIFY LOAD STATUS !
               LDSTAT:
007E 0A09      CPB     RL1,#'9'        ! TEST LEGAL !
0080 3939
0082 E60E      JR      Z,RECACK        ! ACKNOWLEDGEMENTS !
0084 0A09      CPB     RL1,#'0'
0086 3030
0088 E60B      JR      Z,RECACK        ! REC GOOD ACK !
008A 0A09      CPB     RL1,#'7'
008C 3737
008E E608      JR      Z,RECACK


               ! NO ACKNOWLEDGEMENTS RECEIVED !
               ERMSG:
0090 DFD9      CALR    RECMSG          ! GET MCZ MSG !
0092 34CA      LDA     R10,R12(#SNDCHR)
0094 0000*
0096 1FA0      CALL    @R10            ! SEND TO CONS !
0098 0A08      CPB     RL0,#LF
009A 0A0A
009C 9EC6      RET     Z               ! DONE !
009E F8F8      JR      ERMSG

               ! ACKNOWLEDGE RECEIVED !
               RECACK:
00A0 8D43      RESFLG  Z               ! RETURN NZ !
00A2 9E08      RET
00A4           END CMDPAS
```

```
                GLOBAL
                SKPB    LABEL
00A4            SKIPLN  PROCEDURE
                !***************************************
                 *                                     *
                 *  SKIPLN: SKIP RECEIVED LINE FROM    *
                 *          MCZ; RETURN FIRST CHAR OF  *
                 *          NEXT LINE.                 *
                 *                                     *
                 *  REG USE:  RETURN   RL1 = 1ST CHR   *
                 *                     AND NZ IF ESC   *
                 *                                     *
                 ***************************************!
                ENTRY
00A4 DFF3          CALR     RECMSG          ! SKIP OVER LINE !
00A6 0A08          CPB      RL0,#CR         ! THRU CR,LF !
00A8 0D0D
00AA EEFC          JR       NZ,SKIPLN

                SKPB:
00AC 2101          LD       R1,#%3000       ! DELAY FACTOR !
00AE 3000
                ! MAIN LOOP FOR RECEIVING CHAR !
                LOOP1:
00B0 61E0          LD       R0,MCZGET(R14)
00B2 030E
00B4 42E0          CP       R0,MCZPUT(R14)      ! TEST FOR REC CHR !
00B6 030C
00B8 EE03          JR       NZ,RECHR        ! YES,.............. !
00BA AB10          DEC      R1,#1           ! NO, WAIT AWHILE  !
00BC EEF9          JR       NZ,LOOP1
00BE 9E06          RET      Z               ! FORCED EOL !

                RECHR:
00C0 DFFB          CALR     MCZCOM
00C2 0A09          CPB      RL1,#' '        ! CK 1ST=PRNT CHR !
00C4 2020
00C6 9E0D          RET      PL
00C8 DFF5          CALR     RECMSG
00CA E8F0          JR       SKPB
00CC            END SKIPLN
```

```
00CC          MCZCOM PROCEDURE
              !*****************************************
               *                                       *
               *  MCZCOM: LOOPS WAITING FOR RECEIVE    *
               *          CHAR FROM MCZ BY SEEING IF   *
               *          MCZBUF GETS CHAR. DOES       *
               *          ADVANCE POINTER.             *
               *                                       *
               *  REG USE: RETURN  RL1 = CHR           *
               *                                       *
               *****************************************!
              ENTRY
00CC 61E0        LD       R0,MCZGET(R14)    ! CHECK MCZBUF !
00CE 030E
00D0 4BE0        CP       R0,MCZPUT(R14)    ! POINTERS !
00D2 030C
00D4 E6FB        JR       Z,MCZCOM      ! WAIT.....!
00D6 34CA        LDA      R10,R12(#GMCZAD)
00D8 0000*
00DA 1FA0        CALL     @R10          ! GET CHAR FROM BUF !
00DC 2029        LDB      RL1,@R2
00DE 9E08        RET
00E0          END MCZCOM

00E0          RECMSG PROCEDURE
              !*****************************************
               *                                       *
               *  RECMSG: LOOPS WAITING FOR REC CHR    *
               *          FROM MCZ. GETS CHAR AND      *
               *          DO NOT ADVANCE BUF PTR.      *
               *                                       *
               *  REG USE:  RETURNS   RL0 = CHR        *
               *                                       *
               *****************************************!
              ENTRY
00E0 61E0        LD       R0,MCZGET(R14)
00E2 030E
00E4 4BE0        CP       R0,MCZPUT(R14)    ! CK FOR REC !
00E6 030C
00E8 E6FB        JR       Z, RECMSG     ! WAIT..... !
00EA 34CA        LDA      R10,R12(#GMCZAD)
00EC 0000*
00EE 1FA0        CALL     @R10          ! GET 1ST CHAR !
00F0 6FE0        LD       MCZGET(R14),R0    ! RESTORE PTR !
00F2 030E
00F4 2028        LDB      RL0,@R2       ! RTN CHAR !
00F6 9E08        RET
00F8          END RECMSG
```

```
                GLOBAL
                OUTSTM    LABEL
00F8            OUTLNE    PROCEDURE
                !*****************************************
                 *                                       *
                 *  OUTLNE: OUTPUTS A LINE OF CHAR FROM  *
                 *          OUTBUF TO MCZ WITH CR AT     *
                 *          END.                         *
                 *                                       *
                 *  OUTSTM: OUTPUTS A LINE OF CHAR W/CR  *
                 *                                       *
                 *****************************************!
                ENTRY
00F8 61E2          LD       R2,OUTPTR(R14)
00FA 0306
00FC 0C25          LDB      @R2,#CR        !STORE CR IN BUF !
00FE 0D0D
0100 69E0          INC      OUTPTR(R14),#1      ! INC PTR !
0102 0306

                ! NO CR ENTRY POINT !
                OUTSTM:
0104 76F1          LDA       R1,OUTBUF(R14)
0106 2080
                ! MAIN LOOP !
                OVRAGN:
0108 2018          LDB      RL0,@R1
010A A910          INC      R1
010C 34CA          LDA      R10,R12(#SNDMCZ)
010E 0000*
0110 1FA0          CALL     @R10           ! SND CHR TO MCZ !
0112 E603          JR       Z,FINIS
0114 4BE1          CP       R1,OUTPTR(R14)
0116 0306
0118 E7F7          JR       C,OVRAGN       ! CK IF BUF EMPTY !

                ! FINISHED, RESET OUTPTP(R14) AND BLANK OUTBUF !
                FINIS:
011A 76E2          LDA      R2,OUTBUF(R14) ! RESET POINTER !
011C 0080
011E 6FE2          LD       OUTPTR(R14),R2
0120 0306
0122 2100          LD       R0,#OUTSIZ/2
0124 0040
0126 AB00          DEC      R0,#1          ! SET COUNT !
0128 4DF5          LD       OUTBUF(R14),#' '    ! LOAD COUNT !
012A 0080
012C 2020
012E 76F2          LDA      R2,OUTBUF(R14)
0130 0080
```

```
0132 A121       LD      R1,R2
0134 A911       INC     R1,#2
0136 BB21       LDI     @R1,@R2,R0      ! CLR BUFFER !
0138 0010
013A 9F08       RET
013C            END OUTLNE

                ABORTM  LABEL
                GODPAK  LABEL
013C            BADPAK  PROCEDURE
                !*****************************************
                *                                       *
                *  BADPAK: SENDS RESEND SIGNAL ('7')    *
                *          TO MCZ FOR BAD CKSUM OR REC  *
                *          NON-ASCII CHR.               *
                *                                       *
                *  ABORTM: SENDS ABORT SIGNAL ('9')     *
                *          WHEN USER SELECTED.          *
                *                                       *
                *  GODPAK: SENDS ACK SIGNAL ('0') FOR   *
                *          RECEIPT OF GOOD PACKET.      *
                *                                       *
                *****************************************!
                ENTRY
013C C837       LDB     RL0,#'7'        ! LD RESEND SIG !
013E F803       JR      OUTALL
                ABORTM:
0140 C839       LDB     RL0,#'9'        ! LD ABORT SIG !
0142 F801       JR      OUTALL
                GODPAK:
0144 C830       LDB     RL0,#'0'        ! LD REC OK SIG !
                OUTALL:
0146 6EE8       LDB     OUTBUF(R14),RL0
0148 0080
014A 76ED       LDA     R13,OUTBUF(R14)
014C 0090
014E A9D0       INC     R13,#1
0150 6FED       LD      OUTPTR(R14),R13
0152 0306
0154 D02F       CALR    OUTLNE          ! SEND MCZ SYSTEM !
0156 D05A       CALR    SKIPLN          ! SKIP ECHO !
0158 9F08       RET
015A            END BADPAK
```

```
                GLOBAL
015A            GETACK   PROCEDURE
                !*****************************************
                *                                       *
                *  GETACK: RECEIVE AND INTERPRET ACK    *
                *          FROM MCZ. GOOD ACK = '0'     *
                *                    BAD  ACK = '7'     *
                *                    ABORT    = '9'     *
                *                                       *
                *  REG USE:  RETURN  Z,NC IF GOOD ACK   *
                *                    NZ,NC IF BAD ACK   *
                *                    NZ,C IF ABORT      *
                *                                       *
                *****************************************!
                ENTRY
015A D048          CALR     MCZCOM          ! GET CHR !
015C 0A09          CPB      RL1,#'0'        ! CK FOR ACK !
015E 3030
0160 EE04          JR       NZ,NACK         ! NO..... !
0162 D060          CALR     SKIPLN          ! YES, REC ACK !
0164 8D41          SETFLG   Z
0166 8D83          RESFLG   C
0168 9E08          RET

                ! CK FOR '7' AND '9' NON-ACKNOWLEDGEMENTS !
                NACK:
016A 0A09          CPB      RL1,#'7'        ! CK FOR RESEND !
016C 3737
016E EE04          JR       NZ,ABRT         ! NO.... !
0170 D067          CALR     SKIPLN
0172 8D43          RESFLG   Z
0174 8D83          RESFLG   C
0176 9E08          RET

                ! CHECK FOR ABORT !
                ABRT:
0178 0A09          CPB      RL1,#'9'
017A 3939
017C E602          JR       Z,ENDIT         ! YES, ABORT... !
017E D050          CALR     RECMSG          ! GET ANOTHER CHR !
0180 E8EC          JR       GETACK          ! TRY AGAIN.... !

                ENDIT:
0182 D070          CALR     SKIPLN
0184 8D43          RESFLG   Z
0186 8D81          SETFLG   C
0188 9E08          RET
018A            END GETACK
```

```
018A            LINRCT PROCEDURE
                !*****************************************
                 *                                       *
                 *  LINRCT: RECEIVES LINE OF CHAR FROM   *
                 *          MCZ AFTER RECEIPT OF '/',    *
                 *          AND STORES IN INTBUF, ADDING *
                 *          CR AT END AND FILTERING OUT  *
                 *          CONTROL CHARACTERS. (<20H)   *
                 *          (TRUNCATES AFTER 80 CHAR)    *
                 *                                       *
                 *****************************************!
                ENTRY
                ! WAIT FOR ASCII / !
018A D056           CALR    RECMSG
018C 0A08           CPB     RL0,#'/'
018E 2F2F
0190 FEFC           JR      NZ,LINRCT     ! WAIT !
                ! BEGIN STORING CHARACTERS !
0192 76F4           LDA     R4,INTBUF(R14)
0194 0000
0196 CB50           LDB     RL3,#80       !SET LINE LENGTH !
                ! STORE CHAR IN INTBUF !
                LOPSTR:
0198 D05D           CALR    RECMSG        ! GET CHAR !
019A 2E48           LDB     @R4,RL0       ! STORE !
019C 0A08           CPB     RL0,#CR       ! CK FOR END !
019E 0D0D
01A0 FE02           JR      NZ,SKPSOM     ! GOT CHAR.. !
01A2 D07C           CALR    SKPB
01A4 9E08           RET

                !CONTROL CHAR FILTERED AND DEC LINE COUNTER !
                SKPSOM:
01A6 0A08           CPB     RL0,#' '
01A8 2020
01AA E7F6           JR      C,LOPSTR
01AC A940           INC     R4,#1         ! GOOD CHAR !
01AE FB0C           DBJNZ   RL3,LOPSTR    ! DEC COUNT !

                !TRUNCATE, TOO MANY CHAR !
                LOPOVR:
01B0 D069           CALR    RECMSG
01B2 0A08           CPB     RL0,#CR       ! LOOK FOR CR !
01B4 0D0D
01B6 FEFC           JR      NZ,LOPOVR
01B8 76ED           LDA     R13,INTBUF(R14)
01BA 0000
01BC 010D           ADD     R13,#80
01BE 0050
01C0 2ED8           LDB     @R13,RL0
```

```
01C2 9E08       RET
01C4            END LINRCT

01C4            UNPACK   PROCEDURE
                !*****************************************
                *                                       *
                *  UNPACK: UNPACKS RECEIVED PACKETS     *
                *          FROM MCZ IN INTBUF AND       *
                *          LOADS IN SPECIFIED MEMORY    *
                *          AREA. ASCII CHAR ARE CON-    *
                *          VERTED TO HEX VALUES.        *
                *                                       *
                *  REG USE: INPUT  RH3 = #BYTE DATA     *
                *****************************************!
                ENTRY
01C4 A03C       LDB     RL4,RH3         ! SAVE COUNT !
01C6 DFD9       CALR    CONVAD          ! CONV START ADR !

01C8 61E2       LD      R2,DWN_ADR(R14)
01CA 0316
01CC 8D24       TEST    R2
01CE EE02       JR      NZ,NOT_FIRST
01D0 6FE1       LD      DWN_ADR(R14),R1
01D2 0316
                NOT_FIRST:

                ! CHECK FOR USER ENTERED ADDR FOR LOAD !
01D4 0B0B       CP      R11,#%FFFE
01D6 FFFE
01D8 F601       JR      Z,USE_MCZADR
01DA A1B1       LD      R1,R11          ! USER SPECIFIED !

                USE_MCZADR:
01DC 76E2       LDA     R2,INTBUF(R14)
01DE 0000
01E0 A927       INC     R2,#8

                CANDS:
01E2 DFF8       CALR    TENHEX      ! CONVERT 2-ASCII CHR !
01E4 2E18       LDB     @R1,RL0     ! STORE IN MEM !
01E6 A910       INC     R1,#1
01E8 FC04       DBJNZ   RL4,CANDS   ! CONV AND STORE ALL !

                ! UPDATE USER SPECIFIED ADDRESS !
01EA 0B0B       CP      R11,#%FFFE
01EC FFFE
01EE E601       JR      Z,NO_UPDATE   ! USE MCZ ADR !
01F0 A11B       LD      R11,R1        ! UPDATE USER ADR !
```

```
                NO_UPD'TE:
01F2 9E08           RET

01F4            END UNPACK

01F4            TRNHEX PROCEDURE
                !*****************************************
                *                                       *
                *  TRNHEX: CONVERTS TWO ASCII CHAR FRM  *
                *          INTBUF TO TWO 4-BIT HEX #    *
                *          AND ADD TO CKSUM.            *
                *                                       *
                *  REG USE: INPUT  R2 = PTR TO 1ST CHR  *
                *                  RL3= CKSUM ACCUM     *
                *           RETURN R2 = UPDATE PTR      *
                *                  RL3= UPDATED ACCUM   *
                *                  RL0= HEX VALUE       *
                *                  AND C IF NON-ASCII   *
                *                     NC IF ALL GOOD    *
                *                                       *
                *****************************************!
                ENTRY
01F4 DFF6           CALR    ATOHEX          ! CONVERT 1ST CHR !
01F6 9E07           RET     C
01F8 808B           ADDB    RL3,RL0         ! ADD TO CKSUM !
01FA B309           SLA     R0,#12          ! MOVE TO H NIBBLE !
01FC 000C
01FE DFFB           CALR    ATOHEX          ! CONVERT 2ND CHR !
0200 9E07           RET     C
0202 808B           ADDB    RL3,RL0
0204 8408           ORB     RL0,RH0         ! COMBINE NIBBLES !
0206 8D83           RESFLG  C
0208 9E08           RET
020A            END TRNHEX

020A            ATOHEX PROCEDURE
                !*****************************************
                *                                       *
                *  ATOHEX: CONVERTS ONE ASCII CHAR TO   *
                *          4-BIT HEX NIBBLE.            *
                *                                       *
                *  REG USE: INPUT  R2 = PTR TO CHR      *
                *           RETURN R2 = PTR + 1         *
                *                  RL0= HEX NIBBLE      *
                *                                       *
                *****************************************!
                ENTRY
020A 2028           LDB     RL0,@R2
020C A920           INC     R2,#1           ! INC PTR !
020E 34CA           LDA     R10,R12(#CONVERT)
```

```
0210 0000*
0212 1F00      CALL     @R10
0214 9E08      RET
0216           END ATOHEX

0216           CONVAD PROCEDURE
               !*****************************************
                *                                       *
                *  CONVAD: CONVERTS STARTING ADDRESS    *
                *          OF PACKET DATA TO HEX #.     *
                *                                       *
                *  REG USE: RETURN   R1 = ADDRESS(HEX)  *
                *                                       *
                *****************************************!
               ENTRY
0216 76E2      LDA       R2,INTBUF(R14)
0218 0000
021A D014      CALR      TRNHEX
021C A081      LDB       RH1,RL0       ! STORE 1ST BYTE !
021E D016      CALR      TRNHEX
0220 A089      LDB       RL1,RL0       ! STORE 2ND BYTE !
0222 9E08      RET
0224           END CONVAD

0224           CHKPAK   PROCEDURE
               !*****************************************
                *                                       *
                *  CHKPAK: CK RECEIVED MCZ PAC CKSUM    *
                *          AGAINST ACCUMULATED HEX      *
                *          VALUE CKSUM AFTER ASCII-TO-  *
                *          HEX CONVERSION.              *
                *                                       *
                *  REG USE:  RETURN  RH3 = BYTE COUNT   *
                *                    AND C IF BAD OR    *
                *                    NON-ASCII.         *
                *                                       *
                *****************************************!
               ENTRY
0224 76E2      LDA       R2,INTBUF(R14)
0226 0000
0228 C303      LDB       RH3,#3
022A DFF9      CALR      CHKSUM          ! CK 1ST CKSUM !
022C 9E07      RET       C               ! BAD CK !
022E 8C34      TESTB     RH3
0230 9E06      RET       Z               ! NO DATA !
0232 93F3      PUSH      @R15,R3         ! SAVE BYTE COUNT !
0234 DFFE      CALR      CHKSUM          ! CK 2ND CKSUM !
0236 97F3      POP       R3,@R15
0238 9E08      RET
023A           END CHKPAK
```

```
023A            CHKSUM   PROCEDURE
                !*****************************************
                 *                                       *
                 *  CHKSUM: CONVERTS ALL REC ASCII CHR   *
                 *          IN PAC TO HEX AND ACCUM NEW  *
                 *          CKSUM. COMPARE CKSUMS AND    *
                 *          REPORT DIFFERENCES.          *
                 *                                       *
                 *  REG USE: INPUT  R2 = PTR TO PAC      *
                 *                  RH3= # CHR PAIRS     *
                 *           RETURN RH3= BYTE COUNT      *
                 *                  RL3= NEW CKSUM       *
                 *                  RH3= REC CKSUM       *
                 *                  AND C IF BAD OR      *
                 *                  NON-ASCII REC        *
                 *                                       *
                 *****************************************!
                ENTRY
023A 8CB8          CLRB     RL3            ! RESET CKSUM !
023C D025       AB:CALR     TRNHEX         ! CONVERT PAIRS !
023E 9E07          RET      C
0240 F3F3          DBJNZ    RH3,AB         ! CONTINUE..... !
0242 A0B3          LDB      RH3,RL0
0244 93F3          PUSH     @R15,R3        ! SAVE BYTE CNT !
0246 D02A          CALR     TRNHEX         ! CONVERT NEXT TWO !
0248 97F3          POP      R3,@R15
024A 9E07          RET      C
024C 9AB8          CPB      RL0,RL3        ! COMPARE CKSUMS !
024E 9F06          RET      Z              ! GOOD CK... !
0250 9D81          SETFLG   C              ! BAD CKSUM !
0252 9E08          RET
0254            END CHKSUM
```

```
                GLOBAL
0254            LOADFL PROCEDURE
                !*****************************************
                 *                                       *
                 *  LOADFL: RECEIVES PACKET FROM MCZ IN  *
                 *          FOLLOWING FORMAT:            *
                 *                                       *
                 *  <ADR><CNT><CKS1><DTA>...<DTA><CKS2>  *
                 *                                       *
                 *  ADR = START ASR IN Z8000 MEM         *
                 *  CNT = # DATA WORDS                   *
                 *  CKS1= CKSUM OF <ADR> + <CNT>         *
                 *  <DTA>...<DTA> = 32 DATA WORDS        *
                 *  CKS2= CKSUM OF DATA HEX VALUES       *
                 *                                       *
                 *  PROCEDURE VERIFIES CKSUMS BEFORE     *
                 *  STORING DATA IN Z8000 MEM. PACKETS   *
                 *  ARE ACK FOR WITE: '0' = GOOD         *
                 *                    '7' = RESEND       *
                 *                    '9' = ABORT        *
                 *  IF REC '//' FROM MCZ, ECHOS WHAT     *
                 *  REC NEXT TO CONSOLE AND ABORT.       *
                 *                                       *
                 *****************************************!

                ENTRY
0254 4DF8          CLR      DWN_ADR(R14)   ! SET ENTRY ADR BLOCK !
0256 0316
0258 D12D          CALR     FNAME          ! CK FILENAME !
025A 65E4          SET      MFLAGS(R14),#LDMDE !SIGNAL LOAD IN !
025C 0310                                     !PROGRESS!
025E D111          CALR     CMDPAS         ! SND CMD TO MCZ !
0260 9E06          RET      Z              ! Z80 PROG NO LOAD !
                PECLOP:
0262 D06D          CALR     LINRCT         ! GET PACKET !
0264 76E2          LDA      R2,INTBUF(R14)
0266 0000
0268 2028          LDB      RL0,@R2
026A 0A08          CPB      RL0,#'/'       ! CK FOR '//' !
026C 2F2F
026E 7E10          JR       NZ,CONTIN      !NO, CONTINUE...!
0270 76E1          LDA      R1,OUTBUF(R14)   !YES,!
0272 0080
0274 2103          LD       R3,#%20
0276 0020
0278 BB21          LDIR     @R1,@R2,R3     !ERROR MSG SETUP !
027A 0310
027C 76E1          LDA      R1,OUTBUF(R14)
027E 0080
```

```
0280 0101       ADD     R1,#%20
0282 0020
0284 6FE1       LD      OUTPTR(R14),R1 !SET OUTPTR !
0286 0306

0288 34CA       LDA     R10,R12(#PBUFNC)
028A 0000*
028C 1FA0       CALL    @R10
028E 9E08       RET

                CONTIN:
0290 67E5       BIT     MFLAGS(R14),#ESC   ! CK FOR ABORT !
0292 0310
0294 EE34       JR      NZ,ABT             ! YES, ABORT...!
0296 D03A       CALR    CHKPAK             ! CK CKSUMS !
0298 EF02       JR      NC,GDLD            ! GOOD LOAD !
029A DCB0       CALR    BADPAK             ! SEND NON-ACK !
029C F8E2       JR      RECLOP             ! TRY AGAIN !

                ! CHECK FOR LAST PACKET AND PRINT <ENT ADR> !
                GDLD:
029E 8CF8       CLRB    RL3
02A0 8138       ADD     R8,R3    ! ACCUM NUMBER BYTES !
                                 ! OF TRANSFER !

02A2 8C34       TESTB   RH3              ! CK COUNT=0 !
02A4 EE28       JR      NZ,STOR          ! OK, BEGIN STR !
02A6 D0B2       CALR    GODPAK           ! SEND GOOD ACK !
02A8 54E0       LDL     RR0,INTBUF(R14)
02AA 0000
02AC 76ED       LDA     R13,OUTBUF(R14)
02AE 0780
02B0 010D       ADD     R13,#%0C
02B2 000C

                ! CHECK FOR USER SPECIFIED ADDR !
02B4 0B09       CP      R9,#%AAAA
02B6 AAAA
02B8 F61D       JR      Z,END_LOAD   ! NO ECHO TO CONS !
02BA 0B0B       CP      R11,#%FFFE   ! CK FOR LOAD ADR !
02BC FFFE
02BE F6C8       JR      Z,SAME_ADR   ! USE MCZ ADR !
02C0 6FED       LD      OUTPTR(R14),R13  ! SET OUTBUF ADR !
02C2 0306
02C4 61E5       LD      R5,ADR_STR(R14) ! GET USER ADR !
02C6 0314
02C8 76C4       LDA     R10,CONVW(R12)
02CA 0000*
02CC 1FA0       CALL    @R10        ! CONVERT TO ASCII AND !
                                    ! AND STORE IN OUTBUF !
```

```
02CF E801       JR      FIN_BUF
                SAME_ADR:
02D0 1DD0       LDL     @R13,RR0

                FIN_BUF:
02D2 3402       LDAR    R2,ENTADR    !LOAD ENTRY LABEL!
02D4 0040
02D6 76E1       LDA      R1,OUTBUF(R14)
02D8 0080
02DA 2100       LD      R0,#6
02DC 0006
02DE BB21       LDIR    @R1,@R2,R0
02E0 0010
02E2 76ED       LDA     R13,OUTBUF(R14)
02E4 0080
02E6 010D       ADD     R13,#%10
02E8 0010
02EA 6FED       LD      OUTPTR(R14),R13
02EC 0306
02EE 34CA       LDA     R10,R12(#PRNTBF)
02F0 0000*
02F2 1FA0       CALL    @R10          ! PRINT MESSAGE !
                END_LOAD:
02F4 9F08       RET

                STOR:
02F6 D071       CALR    CONVAD
02F8 D0DB       CALR    GODPAK        ! SEND ACK !
02FA D09C       CALR    UNPACK        ! UNPACK AND STORE !
02FC E8B2       JR      RECLOP        ! CONTINUE....!

                ABT:
02FE 3402       LDAR    R2,EMSG
0300 000A
0302 34CA       LDA     R10,R12(#SNDMSG)
0304 0000*
0306 1FA0       CALL    @R10           ! SEND MESSAGE !
0308 D0E5       CALR    ABORTM         ! SEND ABORT !
030A 9E08       RET

030C            END LOADFL

                EMSG:
030C 07         BVAL    7
030E 2F41       WVAL    '/A'
0310 424F       WVAL    'BO'
0312 5254       WVAL    'RT'
0314 0D         BVAL    %0D
                ENTADR:
0316 454E       WVAL    'EN'
```

```
0318 5452      WVAL  'TR'
031A 5920      WVAL  'Y '
031C 504F      WVAL  'PO'
031F 494E      WVAL  'IN'
0321 5420      WVAL  'T '

               END SUPPORT3A
```

APPENDIX E - Support Programs

A. TEXT FILE TRANSFERS

To transfer test files from the MCZ microcomputer RIO system, use the following procedures.

1. Bootup SASS monitor program as described in Appendix C.

2. Bootup INTEL MDS system with CP/M disk having Z8XFER program.

3. Connect cable to MDS system TTY port and to the SASS 'B' connector (to replace line printer).

4. Enter 'transparent' mode of SASS monitor operation to operate within the MCZ RIO operating system, by the following action:

   TYPE "O <CR>"
   (displays RIO prompt '%')

5. Setup to transfer text file by:

   TYPE "PRINT <Filename>"
   (Note: no <CR>)

6. On the MDS system, execute Z8XFER program selecting text file transfer ('T').

7. On SASS (RIO) system, type <CR> to start transfer.

After the entire file has been displayed on SASS terminal, depress any key on MDS terminal to end transfer.

```
'.  Z8000/CPM TRANSFER MODULE


Z8XFER:
        PROCEDURE OPTIONS(MAIN);
        %INCLUDE 'DIOMOD.DCL';
        %REPLACE
                TRUE    BY      '1'B,
                BUFBYT  BY      8,
                FALSE   BY  '0'B;


        DCL
                ANSWER    CHAR(1),
                FN        CHAR(11) VAR,
                I         FIXED,
                WAIT       BIT(1),
                MEM       FIXED(15),
                NUMBUF    FIXED(15),
                REPLY     CHAR(1),
                CMDBUF    CHAR(80) VAR,
                C         CHAR(1),
                CPCNT     FIXED(15),
                TMP       FIXED(15),
                ACCUM     EXT,
                CH1         EXT,
                CH2         EXT,
                PT1       FIXED(15),
                PT2       FIXED(15),
                DFX       FIXED(15),
                ODDEVN    BIT(1),
                ANSR       EXT,
                HEXVAL     EXT,
                OBJXFR    ENTRY    EXT,
                TXTXFR    ENTRY    EXT;

        /*****************************************************
         *     PROGRAM INTERACTIVE HEADER SECTION  *
         *****************************************************/

        PUT SKIP LIST('Z8XFER: SASS TO CPM ASCII-HEX TRANSFER PROGRAM');
        PUT SKIP LIST('    *** NOTE:');
        PUT SKIP LIST('          CABLE CONNECTIONS BETWEEN THE');
        PUT SKIP LIST('          SYSTEMS VARY AS TO TYPE OF');
        PUT SKIP LIST('          FILE BEING TRANSFERED');
        PUT SKIP(2) LIST('             TYPE <CR> TO CONTINUE..');
```

```
/*************************************************************
 *     DETERMINE TRANSFER TYPE                               *
 *************************************************************/

PUT SKIP(2) LIST('IS FILE A TEXT OR CODE FILE? (T/C)');
WAIT=TRUE;
DO WHILE (WAIT=TRUE);
        GET LIST(ANSWER);
        IF (ANSWER = 'T') THEN
                CALL TXTXFR;
        ELSE
        IF (ANSWER = 'C') THEN
                CALL OBJXFR;
        ELSE
                PUT SKIP LIST('INVALID ENTRY');
END;
END Z8XFER;
```

2. TXTXFR MODULE

```
TXTXFR:
        PROC;
        %INCLUDE 'DIOMOD.DCL';
        %REPLACE
                TRUE    BY      '1'B,
                BUFBYT  BY      6,
                FALSE   BY      '0'B;


        DCL
                ANSWER    CHAR(1) EXT,
                FN        CHAR(11) VAR EXT,
                I         FIXED EXT,
                WAIT       BIT(1) EXT,
                MEM       FIXED(15) EXT,
                NUMBUF    FIXED(15) EXT,
                REPLY     CHAR(1) EXT,
                CMDBUF    CHAR(80) VAR EXT,
                C         CHAR(1) EXT,
                CRCNT     FIXED(15) EXT,
                TMP       FIXED(15) EXT,
                ACCUM       EXT,
                CH1        CHAR(1) EXT,
                CH2        CHAR(1) EXT,
                PT1       FIXED(15) EXT,
                PT2       FIXED(15) EXT,
                DEX       FIXED(15) EXT,
                HEXVAL    BIT(8)  EXT,
                ODDEVN    BIT(1) EXT,
                RECTTY    ENTRY   EXT;


        /* TERMINATE TRANSFER AND SAVE THE FILE */
TERMINATE:
        PROC;
        NUMBUF = DIVIDE(MEM,BUFBYT,15);

        /* WRITE BUFFERS TO DISK FILE */
        IF MEM=0 THEN
                DO;
                PUT SKIP LIST('NO DATA TRANSFERED');
                CALL DELETE(ADDR(DESTFILE));
                CALL REBOOT();
                END;
        MEM=0;
        DO I=0 TO NUMBUF-1;
                CALL SETDMA(ADDR(MEMORY(MEM)));
```

```
                MEM=MEM + BUFBYT;
                IF WRSEQ ADDR(DESTFILE) ^=0 THEN
                        DO;
                        PUT SKIP LIST('DISK FULL');
                        CALL REBOOT();
                        END;
        PUT SKIP LIST('TRANSFER COMPLETE.');
        CALL REBOOT();
        END;
        /*************************************************************
         *      TXTXFR: PROCEDURE FOR TRANSFERRING A TEXT            *
         *        FILE FROM SASS TO CPM VIA A CABLE FROM             *
         *        THE SASS 'B' CONNECTOR TO THE INTEL TTY            *
         *        PORT. CPM WILL INTERCEPT SASS CRT TEXT             *
         *        DISPLAY FROM THE MCZ EDITOR.                       *
         *************************************************************/

        DCL
                1 DESTFILE BASED(DFCB0()),
%INCLUDE 'FCB.DCL';
        DCL     MEMORY (0:0)    BIT(16) BASED(MEMPTR());

        /*************************************************************
         *      TFILE: PROC TO DETERMINE NAME OF FILE                *
         *************************************************************/

TFILE:
        PROC;
        /* READ FILE NAME */
        PUT SKIP LIST('WHAT IS THE FILENAME?');
        GET LIST(FN);

        /* PROCESS OPTIONAL DRIVE PREFIX */
        I = INDEX(FN,':');
        IF I=0 THEN
                DESTFILE.DRIVE=0;
        ELSE
                DO;
                DESTFILE.DRIVE=1;
                FN = SUBSTR(FN,I+1);
                END;

        /* GET FILENAME AND TYPE */
        I = INDEX(FN,'.');
        IF I=0 THEN
                DO;
                /* NO FILE SPECIFIED, USE '.TXT' */
                DESTFILE.FNAME = FN;
                DESTFILE.FTYPE = 'TXT';
                END;
```

```
ELSE
        DO;
        DESTFILE.FNAME = SUBSTR(FN,1,I-1);
        DESTFILE.FTYPE = SUBSTR(FN,I+1);
        END;
END TFILE;


/* INIT FCB */
DESTFILE.FEXT = 0;
DESTFILE.CREC = 0;

/* OBTAIN FILENAME AND CHECK FOR EXISTING FILE */
CALL TFILE;
IF SEARCH(ADDR(DESTFILE)) ^= -1 THEN
        DO;
        PUT SKIP LIST('DELETE OLD FILE? (Y/N)');
        GET LIST(ANSWER);
        IF (ANSWER = 'Y') THEN
                CALL DELETE(ADDR(DESTFILE));
        ELSE
                CALL REBOOT();
        END;

/* OPEN NEW FILE */
IF MAKE(ADDR(DESTFILE)) = -1 THEN
        DO;
        PUT SKIP LIST('NO DIRECTORY SPACE');
        CALL REBOOT();
        END;

/* COMPUTE BUFFER SPACE */
NUMBUF = DIVIDE(MEMSIZ(),BUFBYT,15);
IF NUMBUF=0 THEN
        DO;
        PUT SKIP LIST('NO BUFFER SPACE');
        CALL REBOOT();
        END;

/* MAIN LOOP CHECKING FOR KB ENTRY OR TTY RECEIVE */
MEM=0;
WAIT=TRUE;
DO WHILE (WAIT=TRUE);
        IF BREAK() = TRUE THEN
                CALL TERMINATE;
        IF RECTTY = TRUE THEN
                DO;
                IF MEM < NUMBUF THEN
                        DO;
                        MEMORY(MEM)=RDRDR();
```

```
                    MEM = MEM + 1;
                    END;
                ELSE
                    DO;
                    PUT SKIP LIST('FILE TOO LARGE');
                    CALL REBOOT();
                    END;
                END;
            END;
        END;
    END TXTXFR;
```

3. OBJXFR MODULE

```
OBJXFR:
PROC;
%INCLUDE 'DIOMOD.DCL';
%REPLACE
        TRUE    BY      '1'B,
        BUFBYT  BY      128,
        FALSE   BY      '0'B;


DCL
        ANSWER   CHAR(1) EXT,
        FN       CHAR(11) VAR EXT,
        I        FIXED EXT,
        WAIT     BIT(1) EXT,
        MEM      FIXED(15) EXT,
        NUMBUF   FIXED(15) EXT,
        REPLY    CHAR(1) EXT,
        CMDBUF   (1:80)  CHAR(1) EXT,
        C        CHAR(1) EXT,
        CR       CHAR(1),
        CRCNT    FIXED(15) EXT,
        TMP      FIXED(15) EXT,
        ACCUM      EXT,
        HEXVAL   BIT(8)  EXT,
        CH1       CHAR(1) EXT,
        CH2       CHAR(1) EXT,
        PT1      FIXED(15) EXT,
        PT2      FIXED(15) EXT,
        DFX      FIXED(15) EXT,
        ODDEVN   BIT(1) EXT,
        RECTTY   ENTRY   EXT,
        ATOHEX   ENTRY   EXT;
```

```
/*************************************************************
*       OBJXFR: PROCEDURE TO TRANSFER CODE FROM              *
*          Z8000 MEMORY TO CPM FILES BY USING THE            *
*          TECTRONIX FORMAT. CONNECTION IS FROM              *
*          INTEL TTY PORT TO SASS CABLE 'A'. CPM             *
*          WILL ASSUME THE ROLE OF MCZ-RIC SYSTEM.           *
*************************************************************/

DCL
        1 HIFILE BASED(PFCB0()),
%INCLUDE 'FCB.DCL';
DCL
        1 LOFILE BASED(PFCB1()),
%INCLUDE 'FCB.DCL';
DCL
        EVNBUF  (2000) BIT(8),
        ODDBUF  (2000) BIT(8),
        RECBUF  (1:80)  CHAR(1),
        FLAG BIT(1);

/* SASS TO CPM CODE FILE TRANSFER BY TECTRONIX FROMAT */
PUT SKIP LIST('CPM WILL ASSUME THE MCZ-RIO ROLE FOR TRANSFER');
PUT SKIP LIST('     SETUP: CONNECT CABLE TO SASS ''A'' CONNECT-');
PUT SKIP LIST('            TOR AND TO INTEL TTY PORT.');
PUT SKIP LIST('     ENTER <CR> WHEN READY...');
GET LIST(REPLY);
CR='
';

/*************************************************************
*          CFILE: PROCEDURE TO DETERMINE THE                 *
*             FILENAME OF THE OBJECT FILES                   *
*************************************************************/

CFILE:
PROCEDURE;
/* READ FILE NAME */
PUT SKIP LIST('WHAT IS THE FILENAME?');
GET LIST(FN);

/* PROCESS OPTIONAL DRIVE PREFIX */
I = INDEX(FN,':');
IF I=0 THEN
        DO;
        HIFILE.DRIVE = 0;
        LOFILE.DRIVE = 0;
        END;
ELSE
        DO;
        HIFILE.DRIVE = 1;
```

```
        LOFILE.DRIVE = 1;
        FN = SUBSTR(FN,I+1);
        END;

/* SET FNAMES AND FTYPES */
I = INDEX(FN,'.');
IF ^(I=0) THEN
        FN = SUBSTR(FN,1,I-1);
HIFILE.FNAME = FN;
LOFILE.FNAME = FN;
HIFILE.FTYPE = 'OBJ';
LOFILE.FTYPE = 'OBJ';
RETURN;
END CFILE;


/*****************************************************************
 *          ECHO: PROC TO ECHO CMD LINE BACK                     *
 *                TO Z8000 MONITOR.                              *
 *****************************************************************/

ECHO:
PROC;
DCL
        NMBR FIXED(8),
        EXTRA FIXED(8);

/* INIT BUFFER PTR */
NMBR=3;
EXTRA=1;

/* GET CMD LINE FROM MONITOR */
WAIT=TRUE;
DO WHILE (WAIT=TRUE);
        C=RDRDR();
        CMDBUF(NMBR)=C;
        NMBR=NMBR+1;
        /* ECHO BACK */
        IF (C=CR) THEN
                DO WHILE (EXTRA^=NMBR);
                        C=CMDBUF(EXTRA);
                        CALL WRPUN(C);
                        WAIT=FALSE;
                END;
        ELSE
                WAIT=TRUE;
END;
RETURN;
END ECHO;
```

```
/*****************************************************************
 *          GETREC: PROC TO GET ONE TEKTRONIX                    *
 *                  FORMATED RECORD.                             *
 *****************************************************************/

GETREC:
PROC;
CRCNT=1;
WAIT=TRUE;
/* CHECK FOR RECEIVE CHAR FROM CON OR TTY */
DO WHILE (FLAG=TRUE);
        IF BREAK()=TRUE THEN
                DO;
                FLAG=FALSE;
                WAIT=FALSE;
                END;
        IF RECTTY=TRUE THEN
                DO;
                C=RDRDR();
                RECBUF(CRCNT)=C;
                IF C=CR THEN
                        FLAG=FALSE;
                CRCNT=CRCNT+1;
                END;
        END;    /* DO WHILE */
RETURN;
END GETREC;


/*****************************************************************
 *          DOLAST: PROC TO SEND FINAL ACK                       *
 *                  TO Z8000 AND SAVE FILES.                     *
 *****************************************************************/

DOLAST:
PROC;
/* CHECK <CKSUM> FOR DATA FIELD */
TMP=ACCUM;
ACCUM=0;
CH1 = RECBUF(7);
CH2 = RECBUF(8);
CALL ATOHEX;
IF ^(TMP=ACCUM) THEN
        DO;
        C='7';
        CALL WRPUN(C);
        RETURN;
        END;
```

```
/* SEND ACK '0' CHAR TO Z8000 */
C='0';
CALL WRPUN(C);

/* SAVE EVEN PROM FILE (HIFILE) */
NUMBUF = DIVIDE(PT1,BUFBYT,15);
IF PT1=0 THEN
        DO;
        PUT SKIP LIST('NO DATA TRANSFERED');
        CALL DELETF(ADDR(HIFILE));
        CALL DELETF(ADDR(LOFILE));
        CALL REBOOT();
        END;
PT1=0;
DO I = 0 TO NUMBUF-1;
        CALL SETDMA(ADDR(EVNBUF(PT1)));
        PT1=PT1 + BUFBYT;
        IF WRSEQ(ADDR(HIFILE))^=0 THEN
                DO;
                PUT SKIP LIST('DISKFULL');
                CALL REBOOT');
                END;
END;

/* SAVE ODD PROM FILE (LOFILE) */
NUMBUF = DIVIDE(PT2,BUFBYT,15);
PT2=0;
DO I = 0 TO NUMBUF-1;
        CALL SETDMA(ADDR(ODDBUF(PT2)));
        PT2 = PT2 + BUFBYT;
        IF WRSEQ(ADDR(LOFILE))^=0 THEN
                DO;
                PUT SKIP LIST('DISK FULL');
                CALL REBOOT();
                END;
END;

WAIT=FALSE;
PUT SKIP LIST('TRANSFER COMPLETE.');
RETURN;
END DOLAST;
```

```
/*************************************************************
 *         STRREC: PROC TO STORE THE RECEIVED                *
 *                 RECORD IN FILE BUFFERS.                   *
 *************************************************************/

STRREC:
PROC;
IF CRCNT=9 THEN
        CALL DOLAST;
ELSE IF CRCNT=41 THEN
        CALL DASSEM;
ELSE
        DO;
        C='7';
        CALL WRPUN(C);
        END;
RETURN;
END;

/*************************************************************
 *         DASSEM: PROC TO DISASSEMBLE ONE RECORD            *
 *                 AND STORE IN PROPER BUFFER.               *
 *************************************************************/

DASSEM:
PROC;
DEX=1;
ACCUM=0;
PT1=1;
PT2=1;

/* OBTAIN <ADDR><COUNT> CKSUM */
DO I=0 TO 2;
        CH1 = RECBUF(DEX);
        CH2 = RECBUF(DEX+1);
        DEX = DEX + 2;
        CALL ATOHEX;
END;

/* OBTAIN <CKSUM> */
TMP=ACCUM;
ACCUM=0;
CH1 = RECBUF(DEX);
CH2 = RECBUF(DEX+1);
DEX = DEX + 2;
CALL ATOHEX;

/* COMPARE CKSUMS AND REPORT ERROR */
IF ^(TMP=ACCUM) THEN
        DO;
```

```
            C='7';
            CALL WRPUN(C);
            RETURN;
            END;

/* DISASSEMBLE 32-BYTE PACKAGE AND STORE */
ACCUM = 0;
DO I=1 TO 15;
            CH1 = RECBUF(DEX);
            CH2 = RECBUF(DEX+1);
            DEX = DEX + 2;
            CALL ATOHEX;
            /* STORE IN PROPER BUFFER */
            IF ODDEVN=TRUE THEN
                        DO;
                        EVNBUF(PT1)=HEXVAL;
                        HEXVAL=0;
                        PT1=PT1-1;
                        ODDEVN=FALSE;
                        END;
            ELSE
                        DO;
                        ODDBUF(PT2)=HEXVAL;
                        HEXVAL=0;
                        PT2=PT2-1;
                        ODDEVN=TRUE;
                        END;
END;

/* COMPARE CKSUMS AND REPORT ERRORS */
TMP=ACCUM;
ACCUM=0;
CH1 = RECBUF(DEX);
CH2 = RECBUF(DEX+1);
CALL ATOHEX;
IF ^(TMP=ACCUM) THEN
            DO;
            C='7';
            CALL WRPUN(C);
            RETURN;
            END;
ELSE
            DO;
            C='0';
            CALL WRPUN(C);
            WAIT=FALSE;
            RETURN;
            END;
END DASSEM;
```

```
/* INIT FCB'S */
HIFILE.FEXT = 0;
LOFILE.FEXT = 0;
HIFILE.CREC = 0;
LOFILE.CREC = 0;

/* OBTAIN FILENAME AND CHECK FOR EXISTING FILES */
CALL GFILE;
IF (SEAR(ADDR(HIFILE))^=-1 ! SEAR(ADDR(LOFILE))^=-1) THEN
        DO;
        PUT SKIP LIST ('DELETE OLD FILES? (Y/N)');
        GET LIST(ANSWER);
        IF (ANSWER ^= 'Y') THEN
                DO;
                CALL DELETE(ADDR(HIFILE));
                CALL DELETE(ADDR(LOFILE));
                END;
        ELSE
                CALL REBOOT();
        END;

/* OPEN NEW FILES */
IF (MAKE(ADDR(HIFILE))=-1 ! MAKE(ADDR(LOFILE))=-1) THEN
        DO;
        PUT SKIP LIST('NO DIRECTORY SPACE');
        CALL REBOOT();
        END;


/*************************************************************
 *          MAIN CODE SEG: LOOK FOR 'B:', ECHO               *
 *          ALL BACK THRU PUN PORT; SEND '9',                *
 *          SEND '0', AND BEGIN STORING-DISASSEM-            *
 *          BLING TEKTRONIX FORMAT TO THE TWO                *
 *          OBJECT FILES.                                    *
 *************************************************************/

CDDEVN = TRUE;
WAIT = TRUE;

/* WAIT FOR 'B:' */
DO WHILE (WAIT=TRUE);
        IF BREAK()=TRUE THEN
                CALL DOLAST;
        IF RECTTY = TRUE THEN
                DO;
                C=RDRDR();
                IF(C='B') THEN
                        DO;
```

```
                        C=RDRDT();
                        IF (C=';') THEN
                                DO;
                                CMDBUF(1)='B';
                                CMDBUF(2)=';';
                                CALL ECHO;
                                WAIT=FALSE;
                                END;
                        ELSE
                                WAIT=TRUE;
                        END;
                END;
        END;/* DO WHILE */

/* RECEIVE RECORDS AND CONVERT FOR STORAGE */
WAIT=TRUE;
DO WHILE(WAIT=TRUE);
        CALL GETREC;
        CALL STRREC;
END;
END OBJXFR;
```

4. Z8LIB MODULE

```
        NAME    'Z8LIB'
        TITLE   'ASM ROUTINE LIB FOR Z8XFER'
;/*****************************************************************
; *                        DECLARATIONS                           *
; *****************************************************************
;
        PUBLIC  RECTTY          ;RETURNS '1' FOR RFC CHAR
        PUBLIC  ATOHEX          ;CONVERTS TWO ASCII BYTES TO ONE
        PUBLIC  POST            ;ERROR SIGNAL
        PUBLIC  ACCUM           ;ACCUMULATOR FOR CKSUM
        PUBLIC  HEXVAL          ;RETURN OF HEX BYTE
        PUBLIC  ANSR            ;MAILBOX FOR PL/I RETURN
        PUBLIC  CH1             ;FIRST ASCII CHAR
        PUBLIC  CH2             ;SECOND ASCII CHAR
;
; *****************************************************************
; *                          EQUATES                              *
; *****************************************************************
;
TPORT   EQU     0F5H    ;TTY (RDR) CMD PORT ADDR
POST    DB      1       ;ERROR FLAG FOR Z8XFER FOR BAD CH
ACCUM   DB      2       ;ACCUM VALUE FORR CHECK SUMS
HEXVAL  DB      1       ;RETURNED HEX BYTE FROM 2-ASCII
ANSR    DB      1
CH1     DB      1       ;PASSED CHAR FROM BUFFERS
CH2     DB      1       ;SAME
;
; *****************************************************************
; *             RECTTY: ROUTINE TO FACILITATE DIRECT              *
; *                     I/O STATUS READ TO TTY PORT.              *
; *****************************************************************
;
RECTTY: ANI     0               ;CLEAR REG A
        IN      TPORT           ;READ TTY CMD PORT STATUS
        ANI     01H             ;CHECK FOR RECEIVE CHAR
        JZ      FINI            ;NO CHAR RECEIVED
        MVI     A,01H           ;YES, CHAR RECEIVED
;
FINI:   MOV     L,A
        MVI     H,0             ;LOAD ANSWER
        SHLD    ANSR            ;STORE IN MAILBOX
        RET
;
;
```

```
;  ***************************************************************
;  *              ATOHEX: ROUTINE TO CONVERT TWO ASCII           *
;  *                      CH INTO ONE HEX BYTE.                  *
;  ***************************************************************
;
ATOHEX: CALL    CONBYT          ;CONVERT FIRST CHAR TO NIBBLE
        CPI     0FFH            ;FIRST BYTE CONVERT ERROR
        JZ      DONE            ;QUIT
        MOV     E,A             ;STORE FIRST NIBBLE
        LDA     CH2
        STA     CH1             ;GET SECOND CHAR
        CALL    CONBYT          ;CONVERT SECOND CHAR TO NIBBLE
        CPI     0FFH            ;SECOND BYTE CONVERT ERROR
        JZ      DONE            ;QUIT
        MOV     C,A             ;TEMP STORE SECOND NIBBLE
        MOV     A,E             ;GET HIGH NIBBLE
        RAL
        RAL
        RAL
        RAL                     ;ROTATE TO HIGH NIBBLE
        ANI     0F0H            ;CLR LOW NIBBLE
        ADD     C               ;FORM HEX BYTE
        STA     HEXVAL          ;RETURN HEX BYTE
DONE:   RET
;
;  ***************************************************************
;  *                                                             *
;  ***************************************************************
;
CONBYT: LDA     CH1             ;GET ASCII CHAR
        CPI     30H             ;NO CHAR <30H
        JC      ERR
        CPI     3AH             ;CK IF DIGIT
        JC      CK              ;YES, IS DIGIT
        CPI     41H             ;CK IF 'A' OR GREATER
        JC      ERR
        CPI     47H             ;CK IF <'F'
        JNC     ERR
CK:     ANI     0FH             ;NEED LOWER NIBBLE
        MOV     E,A             ;ADD TO ACCUMULATOR
        MVI     D,0
        LHLD    ACCUM
        DAD     D               ;ADD NEW BYTE
        SHLD    ACCUM           ;STORE
ERR:    RET
;
;  ***************************************************************
;  *                                                             *
;  ***************************************************************
        END
```

C. CPMXFR PROGRAM LISTING

1. Z8000 TRANSFER MODULE

```
Z8000ASM  2.02
LOC    OBJ CODE    STMT SOURCE STATEMENT

                       1 TRANSFER MODULE
                $LISTON  $TTY

                CONSTANT
                   PORTAD    :=  %FFD9
                   PORTAC    :=  %FFDB
                   PROM_SIZ  :=  %1005
                   MEM_START :=  %6000
                   TXR       :=  0
                   FXFC      :=  %00EC

                $REL 0
                GLOBAL
0000            TRANS  PROCEDURE
                !*****************************************
                 *                                       *
                 *  TRANS: TRANSFERS OBJECT CODE IN      *
                 *         Z8000 MEMORY TO INTEL CP/M    *
                 *         OBJECT FILE ON DISK.          *
                 *                                       *
                 *****************************************!
                ENTRY
0000 2101          LD       R1,#PROM_SIZ   ! SET TRANS LIMIT !
0002 1005
0004 2102          LD       R2,#MEM_START  ! SET MEMORY START !
0006 6000
                                           ! ADDRESS !
0008 8D38          CLR      R3

                ! *** MAIN LOOP SENDING BYTES *** !

                DO
000A 2029          LDB      RL0,@R2    ! GET BYTE FROM MEM !
000C DFF8          CALR     SNDCHR     ! SEND CHAR TO CPM !
000E A920          INC      R2,#1      ! ADVANCE ADR 1 BYTE !
0010 A930          INC      R3,#1      ! INCREMENT COUNTER !
0012 8B13          CP       R3,R1      ! DONE?.... !
0014 F601          JR       Z,FINI     ! YES, DONE !
0016 EFF9       OD

                FINI:
```

```
0018 2101       LD      R1,#TXEC
001A 00FC
001C 1F19       JP      @R1

001E            END TRANS

001E            SNDCHR  PROCEDURE
                !*****************************************
                 *                                       *
                 *  SNDCHR: SENDS SINGLE BYTE TO INTEL   *
                 *          CP/M AFTER STATUS CHECK.     *
                 *                                       *
                 *****************************************!
                ENTRY
001E 3A04       INB     RH0,PORTAC      ! GET STATUS !
0020 FFDB
0022 A600       BITB    RH0,#TXR        ! CK IF TRANSMIT RDY !
0024 F6FC       JR      Z,SNDCHI
0026 3A96       OUTB    PORTAD,RL0      ! SEND BYTE !
0028 FFD9
002A 9E08       RET
002C            END SNDCHR

                END TRANSFER

   0 errors
Assembly complete
```

LIST OF REFERENCES

1. O'Connell, J. S., and Richardson, L. D., Distributed Secure Design for a Multi-Microprocessor Operating System, MS Thesis, Naval Postgraduate School, June 1979.

2. Parks, E. J., The Design of a Secure File Storage System, MS Thesis, Naval Postgraduate School, December 1979.

3. Coleman, A. R., Security Kernel Design for a Microprocessor-Based, Multilevel, Archival Storage System, MS Thesis, Naval Postgraduate School, December 1979.

4. Moore, E. F. and Gary, A. V., The Design and Implemention of the Memory Manager for a Secure Archival Storage System, MS Thesis, Naval Postgraduate School, June 1980.

5. Reitz, S. L., An Implementation of Multiprograming and Process Management for a Security Kernel Operating System, MS Thesis, Naval Postgraduate School, June 1980.

6. Wells, J. T., Implementation of Segment Management for a Secure Archival Storage System, MS Thesis, Naval Postgraduate School, September 1980.

7. Strickler, A.R., Implementation of Process Management for a Secure Archival Storage System, M.S. Thesis, Naval Postgraduate School, March 1981.

8. Organick, E. I., The Multics System: An Examination of Its Structure, MIT Press, 1972.

9. Madnick, S. E., and Donovan, J. J., Operating Systems, McGraw Hill, 1974.

10. Reed, P. D., Processor Multiplexing In a Layered Operating System, MS Thesis, Massachusetts Institute of Technology, MIT LCS/TR-167, 1979.

11. Schell, R.R., Dynamic Reconfiguration in a Modular Computer System, Ph.D. Thesis, Massachusetts Institute of Technology, May 1971.

12. Schell, Lt.Col. R. R., "Security Kernels: A Methodical Design of System Security," USE Technical Papers Spring Conference, 1979). p. 245-250, March 1979.

13. Denning, D. E., "A Lattice Model of Secure Information Flow," Communications of the ACM, v. 19, p. 236-242, May 1976.

14. Schroeder, M. D., "A Hardware Architecture for Implementing Protection Rings," Communications of the ACM, v. 15, no. 3, p. 157-170, March 1972.

15. Dijkstra, E.W., 'The Structure of the 'THE' Multiprogramming System', Communications of the ACM, v. 11, p. 341-346, May 1968.

16. Reed, P. D., and Kanodia, R. K., "Synchronization With Eventcounts and Sequencers," Communications of the ACM, v. 22, no. 2, p. 115-124, February 1979.

17. Luniewski, A., A Simple and Flexible System Initialization Mechanism, M.S. Thesis, Massachusetts Institute of Technology, May 1977.

18. Ross, J.I., Design of a System Initialization Mechanism for a Multiple Microcomputer, M.S. Thesis, Naval Postgraduate School, June 1980.

19. Anderson, E.L., Automatic Recovery in a Real-Time, Distributed Multiple Microprocessor Computer System, M.S. Thesis, Naval Postgraduate School, December 1980.

20. Advance Micro Devices, Am9513 System Timing Controller, Product Specification Sheet, 1980.

21. Zilog, Inc., Z8001 CPU Z8002 CPU, Preliminary Product Specification, March 1979.

22. Zilog, Inc., Z8010 MMU Memory Management Unit, Preliminary Product Specification, October 1979.

23. Advanced Micro Computers, AM96/4116 AmZ8000 16-Bit MonoBoard Computer, User's Manual, 1980.

24. Zilog, Inc., Z8000 PLZ/ASM Assembly Language Programming Manual, 03-3055-01, Revision A, April 1979.

25. Schell, R. R. and Cox, L. A., Secure Archival Storage System, Part I - Design, Naval Postgraduate School, NPS52-80-002, March 1980.

26. Schell, R. R. and Cox, L. A., Secure Archival Storage System, Part II - Segment and Process Management Implementation, Naval Postgraduate School, NPS52-81-001, March 1981.

27. Baker, G. S., Secure Archival Storage System - Hardware Architecture and Developmental Monitor, Naval Postgraduate School, NPS__-81-___, June 1981.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center ATTN:DDC-TC Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93940	2
3.	Department Chairman, Code 52 Department of Computer Science Naval Postgraduate School Monterey, California 93940	2
4.	LTCOL Roger R. Schell, Code 52Sj Department of Computer Science Naval Postgraduate School Monterey, California 93940	2
5.	Lyle A. Cox, Jr., Code 52C1 Department of Computer Science Naval Postgraduate School Monterey, California 93940	6
6.	Joel Trimble, Code 221 Office of Naval Research 800 North Quincy Arlington, Virginia 22217	1
7.	Department Chairman Department of Computer Science United States Military Academy West Point, New York 10996	1
8.	INTEL Corporation Attn: Mr. Robert Childs Mail Code: SC 4-490 3065 Bowers Avenue Santa Clara, California 95051	1
9.	John P.L. Woodward The MITRE Corporation P.O. Box 208 Bedford, Massachusetts 01730	1

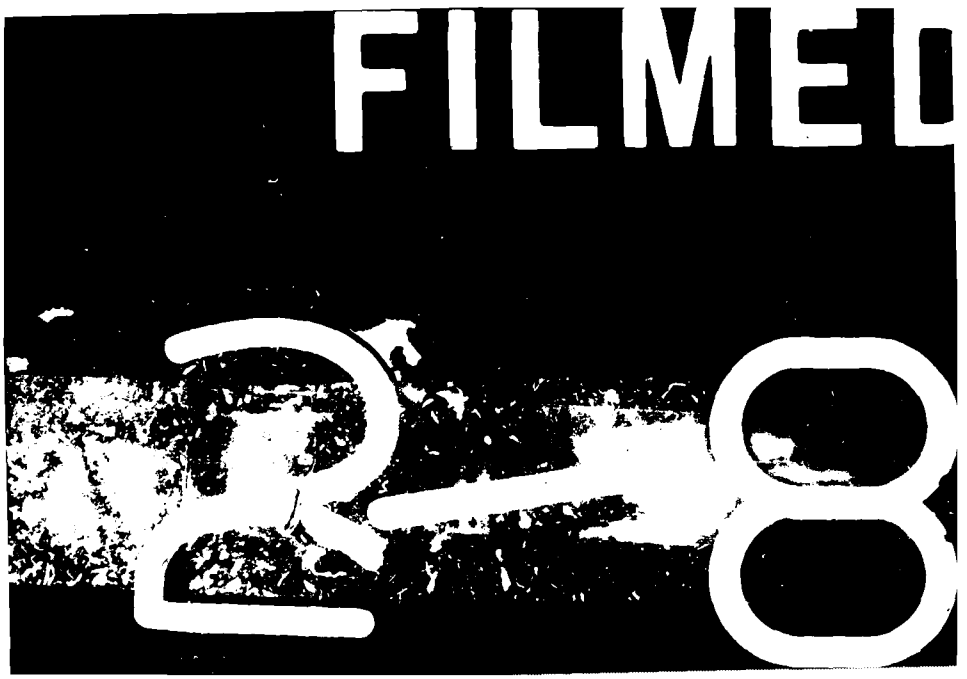
10. Digital Equipment Corporation 1
Attn: Mr. Donald Gaubatz
146 Main Street
ML 3-2/E41
Maynard, Massachusetts 01754

11. Joe Urban 1
University of Southwestern Louisiana
P.O. Box 44330
Lafayette, Louisiana 70504

12. LCDR Gary Baker, Code 37 3
COMRESPATWINGPAC
NAS Moffett Field
Moffett Field, California 90031

13. CPT Anthony R. Strickler 1
Route #12
West Shipley Ferry Road
Kingsport, Tennessee 37663

14. Professor T.F. Tao, Code 62Tv 1
Department of Electrical Engineering
Naval Postgraduate School
Monterey California 93940


FILME
R-8